Universidad de Alcalá

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

ANÁLISIS DE LOS NUEVOS PARADIGMAS DE COMPUTACIÓN

Autor: Sergio Fuentes Izquierdo

Tutor: Pablo Ramos Sainz

2017

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

ANÁLISIS DE LOS NUEVO PARADIGMAS DE COMPUTACIÓN

Autor: Sergio Fuentes Izquierdo

Tutor: Pablo Ramos Sainz

TRIBUNAL:

Presidente: José Manuel Rodríguez Ascariz

Vocal 1º: Fernando Naranjo Vega

Vocal 2º: Pablo Ramos Sainz

CALIFICACIÓN: …….

FECHA: ……

A mi tutor, Pablo Ramos, por toda la ayuda que me ha ofrecido durante estos meses.

A mis compañeros y profesores, por hacer que esta etapa de mi vida sea inolvidable.

A mi madre, por buscar siempre lo mejor, anteponerme en muchas de sus decisiones y darme la educación de la que dispongo.

A mis abuelos, porque sin su ayuda, su esfuerzo y sus valores no sería la persona que soy hoy.

A Laura, por ser mi apoyo en los malos momentos, por darme su amor y llenar mi vida.

ÍNDICE GENERAL

1 – RESUMEN

El presente Trabajo de Fin de Grado trata de un análisis de los nuevos paradigmas de la computación. Para realizar este análisis se han estudiado las tecnologías de la *computación cuántica* y de la *memcomputación*.

El estudio se basa en cinco puntos, que son: una breve introducción de la tecnología, la explicación de las bases de la tecnología, las condiciones necesarias para poder construir una máquina basada en la tecnología, la máquina universal de la tecnología y la resolución de un problema NP-completo con la respectiva tecnología.

Palabras clave: computación cuántica, memcomputación, qubit, memelemento.

2 – ABSTRACT

The present Degree's Final Project is an analysis of the new computing paradigms. To perform this analysis, we have studied the quantum computing and the memcomputing.

This study is based on five points, which are: a brief introduction of the technology, an explanation of the basis of the technology, the necessary conditions to be able to build a machine-based technology, the universal machine of the technology and the resolution of a NP-complete problem with the respective technology.

Key words: quantum computing, memcomputing, qubit, mem-element.

3 – RESUMEN EXTENDIDO

El desarrollo y la evolución de la tecnología han hecho posible que la escala de integración haya aumentado exponencialmente, haciendo que en el mismo espacio quepan más transistores. De esta manera cada vez somos capaces de fabricar chips más pequeños, y es que, cuanto más pequeño es el chip, mayor velocidad de proceso alcanza. Sin embargo, según se reducía el tamaño de los chips se han encontrado problemas de funcionamiento, lo que implica que no se pueden fabricar chips infinitamente pequeños. Cuando se llega a la escala de nanómetros los electrones escapan de los canales por donde deben circular. Esto es conocido como el efecto túnel.

Basándonos en este principio, podemos llegar a la conclusión de que la computación digital tradicional no tardaría en llegar a su límite y surge la necesidad de investigar nuevas tecnologías de computación que permitan la continuidad de la evolución de la tecnología.

Como alternativa a la computación clásica, surgió en los años 80 la idea de *computación cuántica*. Esta alternativa se basa en las *leyes cuánticas*, haciendo que una partícula pueda encontrarse en superposición coherente: puede ser 0, 1 y 0 y 1 al mismo tiempo. Esto permite que se puedan realizar varias operaciones al mismo tiempo.

Aunque esta alternativa es la más conocida y evolucionada, no es la única. Otra alternativa que recientemente ha tomado fuerza es la *memcomputación*. Esta nueva alternativa se inspira en el cerebro humano, donde nuestras neuronas son capaces de almacenar y procesar la información simultáneamente. En los ordenadores convencionales el almacenamiento y el proceso de la información tienen lugar en regiones de espacio distintas, lo que produce limitaciones de espacio y retrasos a la hora de conseguir y procesar información importante. Esta nueva alternativa permitiría corregir estos errores.

4 – CONCEPTOS TEÓRICOS PREVIOS

4.1 – MECÁNICA CUÁNTICA

La *mecánica cuántica* es un marco matemático para el desarrollo de teorías físicas. La *mecánica cuántica* por si misma no dice qué leyes debe obedecer un sistema físico, pero provee un marco matemático y conceptual para el desarrollo de esas leyes.

Los postulados de la *mecánica cuántica* ofrecen una conexión entre el mundo físico y el formalismo matemático de la *mecánica cuántica*. Estos postulados derivan de un largo proceso de ensayo y error.

- **Postulado 1**: Asociado con cada sistema físico, se encuentra un espacio de Hilbert (espacio vectorial complejo con producto interno normalizado) conocido como espacio de estados del sistema. El sistema es completamente descrito por su vector de estado, el cual es un vector unitario en dicho espacio de estados.

- **Postulado 2**: La evolución de un *sistema cuántico* cerrado (estrictamente aislado, sin intercambio de energía con su alrededor) es descrita por una trasformación unitaria. Esto es que el estado $|\Psi\rangle$ del sistema en el instante t_1 está relacionado con el estado $|\Psi'\rangle$ en el instante t_2 mediante un operador unitario *U* que depende solo de t_1 y t_2 tal que $|\Psi'\rangle = U|\Psi\rangle$.

- **Postulado 2'**: La evolución en el tiempo de un *sistema cuántico* cerrado es descrita por la ecuación de onda de Schrödinger $i\hbar\frac{d|\Psi\rangle}{dt} = H|\Psi\rangle$ donde *H* es el operador Hamiltoniano (hermítico y unitario) y $\hbar$ es la constante de Planck.

- **Postulado 3**: Las *mediciones cuánticas* son descritas por un conjunto *{M_n}* de operadores de medición. Estos operadores actúan sobre el espacio de estados del sistema medido. El subíndice *m* se refiere a los resultados posibles del experimento de medición. Si el estado de un *sistema cuántico* es $|\Psi\rangle$ inmediatamente antes de la medición, la probabilidad de ocurrencia de *m* es $p(m) = \langle\Psi|M_m^+ M_m|\Psi\rangle$ y el estado del sistema después de la medición es $\frac{M_m|\Psi\rangle}{\sqrt{\langle\Psi|M_m^+ M_m|\Psi\rangle}}$. Los operadores de medición satisfacen la condición de completamiento $\sum_m M_m^+ M_m = I$.

- **Postulado 4**: El espacio de estados de un sistema físico compuesto es el producto tensorial de los espacios de estado de los componentes. Específicamente, si tenemos los sistemas numerados de *1* a *n*, si el estado del sistema *1* es $|\Psi_1\rangle$ y así sucesivamente, el estado conjunto del sistema total será $|\Psi_1\rangle \otimes |\Psi_2\rangle \otimes \ldots \otimes |\Psi_n\rangle$.

Para tener una perspectiva global de los postulados podemos decir que:

- El primer postulado describe el escenario de la *mecánica cuántica* al fijar el modo de describir los *estados cuánticos*.
- El segundo nos habla de la dinámica de los estados, lo que se resuelve por la ecuación de Schrödinger y los operadores unitarios.
- El tercer postulado nos informa de cómo extraer información de un *sistema cuántico* a través de mediciones.
- El cuarto postulado nos aclara cómo se combinan *estados cuánticos* individuales en un sistema total superpuesto. [1]

4.2 – EL GATO DE SCHÖDINGER

Cuando se habla del “gato de Schrödinger” se está haciendo referencia a una paradoja que surge de un célebre experimento propuesto por Erwin Schrödinger para ilustrar las diferencias entre interacción y medida en el campo de la *mecánica cuántica*.

El experimento consiste en imaginar a un gato metido dentro de una caja que contiene un peligroso dispositivo. Este dispositivo está formado por una ampolla de vidrio que contiene veneno y por un martillo sujeto sobre la ampolla de forma que si cae sobre ella la rompe y escapa el veneno, con lo que el gato moriría. El martillo está conectado a un mecanismo detector de partículas alfa; si llega una partícula alfa el martillo cae rompiendo la ampolla con lo que el gato muere, por el contrario, si no llega no ocurre nada y el gato continua vivo.

Cuando el dispositivo está preparado, se realiza el experimento. Al lado del detector se sitúa un átomo radiactivo con un 50% de probabilidades de emitir una partícula alfa de una hora. Al cabo de una hora habrá ocurrido uno de los dos sucesos posibles: el átomo ha emitido una partícula alfa o no la ha emitido, ambas opciones con la misma probabilidad. Como resultado de la interacción, en el interior de la caja, el gato está vivo o muerto. Pero no podemos saberlo si no la abrimos para comprobarlo.

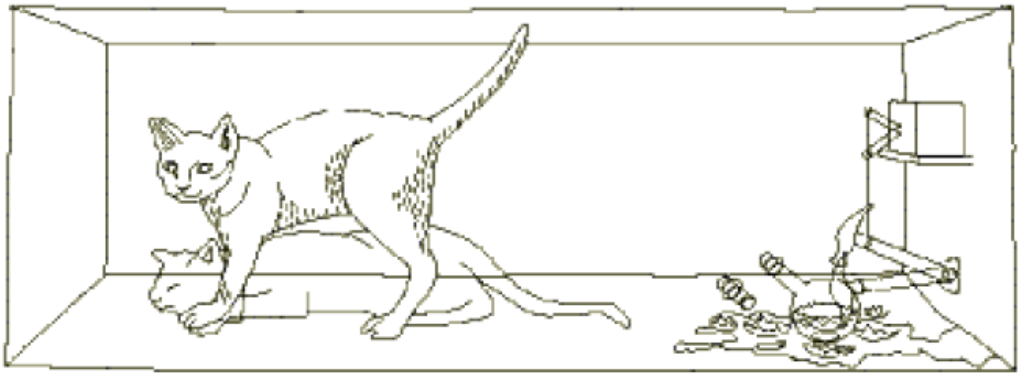

Fig. 1. El gato de Schrödinger.

Si lo que ocurre en el interior de la caja lo intentamos descubrir aplicando las leyes de la *mecánica cuántica*, llegamos a una conclusión muy extraña. El gato estará descrito por una función de onda extremadamente compleja resultado de la superposición de dos estados combinados al cincuenta por ciento: “gato vivo” y “gato muerto”. Es decir, aplicando el *formalismo cuántico*, el gato estará a la vez vivo y muerto; se trataría de dos estados indistinguibles.

La única forma de averiguar qué ha ocurrido con el gato es realizar una medida: abrir la caja y mirar dentro. En unos casos nos encontraremos al gato vivo y en otros muerto. Pero, ¿qué ha ocurrido? Al realizar la medida, el observador interactúa con el sistema y lo altera, rompe la superposición de estados y el sistema se decanta por uno de sus dos estados posibles.

El sentido común nos indica que el gato no puede estar vivo y muerto a la vez, pero la *mecánica cuántica* dice que, mientras nadie mire en el interior de la caja, el gato se encuentra en una superposición de los dos estados.

Esta superposición de estados es una consecuencia de la naturaleza de la materia y su aplicación a la descripción de la *mecánica cuántica* de los sistemas físicos, lo que permite explicar el comportamiento de las partículas elementales y de los átomos. La aplicación a sistemas macroscópicos como el gato nos llevaría a la paradoja que propone Schrödinger. [2]

4.3 – ARQUITECTURA VON NEUMANN

El modelo de Von Neumann define una máquina con cuatro subsistemas: memoria, unidad aritmético lógica, unidad de control y unidad de entrada/salida.

- **Memoria**: La memoria es almacenamiento, donde los programas y los datos se almacenan durante el procesamiento.
  A la cantidad de palabras que forman la memoria se le denomina capacidad de memoria. De este modo, cuanto mayor sea el número de palabras, mayor será el número de instrucciones y datos que podrá almacena la máquina.

- **Unidad aritmético lógica**: La unidad aritmético lógica (ALU) es donde tienen lugar los cálculos y las operaciones lógicas.

- **Unidad de control**: La unidad de control determina las operaciones de la memoria, de la ALU y del subsistema de entrada/salida. Se podría decir que la unidad de control gobierna y coordina las actividades de una máquina.

- **Unidad de entrada/salida**: El subsistema de entrada acepta datos de entrada y el programa desde el exterior de la máquina, mientras que el subsistema de salida envía el resultado del procesamiento al exterior. La definición del subsistema de entrada/salida es muy amplia, ya que también incluye los dispositivos de almacenamiento secundarios. [3]

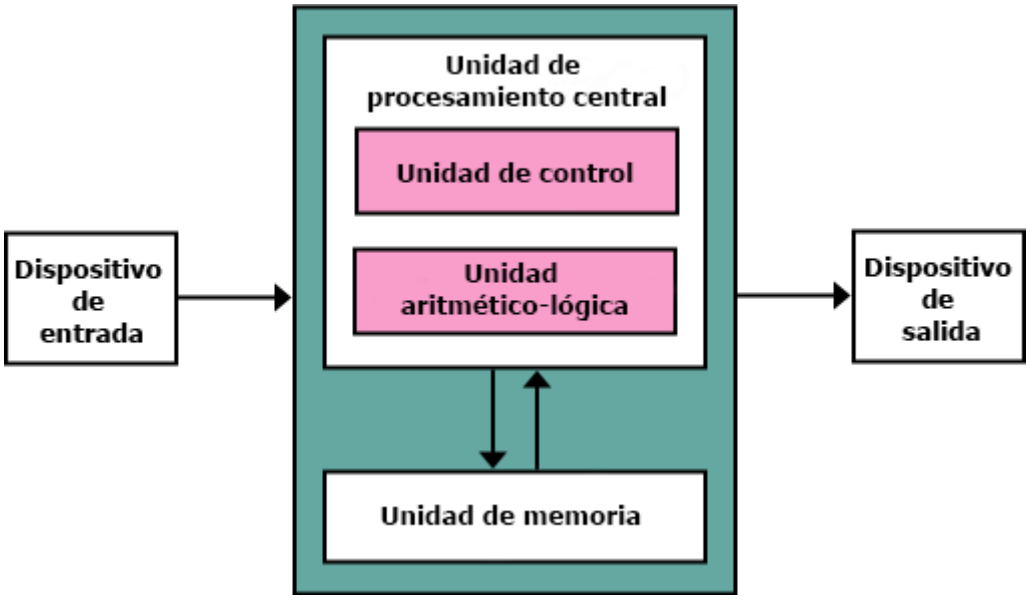

Fig. 2. Diagrama de la arquitectura Von Neumann.

4.4 – EL BIT

Bit es el acrónimo de binary digit. Un bit es un dígito del sistema de numeración binario.

Mientras que en el sistema de numeración decimal se utilizan diez dígitos, en el binario se usan solo dos dígitos, el 0 y el 1. Un bit puede representar unos de esos dos valores, 0 o 1.

El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital o en la teoría de la información. Con él, podemos representar dos valores cualesquiera. Basta con asignar uno de esos valores al estado de "apagado" (0) y el otro al estado de "encendido" (1). [4]

4.5 – PUERTAS LÓGICAS

Una puerta lógica es un dispositivo electrónico con una función booleana, es decir, realizan una operación lógica en uno o más bits de entrada y produce una salida binaria.

NOT	AND	NAND	OR	NOR	XOR	XNOR
$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
						

A	X
0	1
1	0

B	A	X
0	0	0
0	1	0
1	0	0
1	1	1

B	A	X
0	0	1
0	1	1
1	0	1
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	1

B	A	X
0	0	1
0	1	0
1	0	0
1	1	0

B	A	X
0	0	0
0	1	1
1	0	1
1	1	0

B	A	X
0	0	1
0	1	0
1	0	0
1	1	1

Fig. 3. Puertas lógicas.

Las puertas NOR y las puertas NAND se pueden utilizar para reproducir todas las funciones de las otras puertas lógicas. [5]

4.6 – ELEMENTOS BÁSICOS DE CIRCUITOS LINEALES

En los circuitos electrónicos se pueden encontrar dos tipos de elementos: elementos pasivos y elementos activos. Un elemento activo es capaz de generar energía, mientras que un elemento pasivo no puede. Los elementos pasivo fundamentales son el resistor, el condensador y el inductor. [6]

4.6.1 – RESISTOR

Los materiales tienen un comportamiento característico de resistir el flujo de carga eléctrica. Esta propiedad física es conocida como resistencia y se representa con el símbolo *R*, que es la constante proporcional entre tensión y corriente. Esta relación es conocida como la ley de Ohm.

$$v = iR \tag{1}$$

La resistencia *R* puede medirse como

$$R = \rho \frac{l}{A} \tag{2}$$

donde *ρ* es la resistividad del material, *A* es el área de la sección de corte y *l* es la longitud. El consumo de potencia en la resistencia viene dado por [6]

$$P = iv = i^2R = \frac{v^2}{R} \tag{3}$$

4.6.2 – CONDENSADOR

Un condensador es un elemento pasivo diseñado para almacenar energía en su campo eléctrico.

Cuando una fuente de tensión se conecta al condensador, la fuente deposita una carga positiva *q* en un plano y una carga negativa en el otro. El condensador debe almacenar carga eléctrica. La cantidad de carga almacenada, representada por *q*, es directamente proporcional a la tensión aplicada

$$q = Cv \tag{4}$$

donde *C* es la capacitancia del condensador. Aunque la capacitancia *C* es la relación entre la carga almacenada y la tensión aplicada no depende de ninguna de ellas. La capacitancia depende de las dimensiones físicas del condensador

$$C = \frac{\varepsilon A}{d} \tag{5}$$

donde *A* es el área de la superficie de cada plano, *d* es la distancia entre planos y ***ε*** es la permitividad del material dieléctrico entre planos. La relación entre la tensión y la corriente de un condensador viene dada por

$$i = C\frac{dv}{dt} \tag{6}$$

La potencia instantánea proporcionada por el condensador es calcula como

$$p = iv = Cv\frac{dv}{dt} \tag{7}$$

La energía almacenada en el condensador viene dada por

$$w(t) = \int_{-\infty}^{t} p(t)dt = \frac{1}{2}C\left(v^2(t) - v^2(-\infty)\right) \tag{8}$$

Hay que anotar que $v(-\infty) = 0$, porque el condensador se encuentra descargado en $t = -\infty$. Así que [6]

$$U(t) = \frac{1}{2}Cv^2(t) = \frac{q^2}{2C} \tag{9}$$

4.6.3 – INDUCTOR

El inductor es un elemento pasivo diseñado para almacenar energía en su campo magnético.

La tensión a través del inductor es linealmente proporcional a la relación del cambio en la corriente como

$$v = L\frac{di}{dt} \tag{10}$$

donde *L* es la inductancia del inductor. La inductancia depende de la dimensión física y la construcción del inductor. La inductancia de un solenoide se calcula como

$$L = \frac{N^2 \mu A}{l} \tag{11}$$

donde *N* es el número de vueltas, *l* es la longitud, *A* es el área de la sección de corte y *μ* es la permeabilidad del núcleo. La relación entre la tensión y la corriente se obtiene mediante

$$i = \frac{1}{L}\int_{-\infty}^{t} v(t)dt \tag{12}$$

El inductor está diseñado para almacenar energía en su campo magnético. La potencia proporcionada por el inductor en función de la corriente es

$$p = vi = \left(L\frac{di}{dt}\right)i \tag{13}$$

La energía almacenada viene dada por

$$U = L\int_{-\infty}^{t} i(t)dt = \frac{1}{2}Li^2(t) - \frac{1}{2}Li^2(-\infty) \tag{14}$$

la cual tenderá a $w = \frac{1}{2}Li^2(t)$, ya que $i(-\infty) = 0$. [6]

4.7 – MÁQUINA DE TURING

La máquina de Turing es un formalismo matemático descrito por Alan Turing con la finalidad de disponer de una herramienta abstracta para estudiar la teoría de la computabilidad. Se dice que un problema es computable si existe un algoritmo que permita obtener su solución. Turing ideó herramientas cuyo comportamiento se puede describir de manera sencilla, lo que permite estudiar con procedimientos algebraicos la computabilidad de los problemas.

Intuitivamente, una máquina de Turing puede considerarse como un dispositivo capaz de adoptar un estado entre varios posibles y que dispone de un cabezal capaz de leer y escribir símbolos en una cinta. En cada momento, en función del estado en que se encuentra y del símbolo que lee la cinta, realiza tres acciones:

- Escribe un nuevo símbolo en la cinta, en la misma casilla donde acaba de leer un símbolo.

- Desplaza el cabezal una única posición a lo largo de la cinta o bien se puede dejar apuntando a la misma casilla.

- Cambia de estado. [7]

4.8 – COMPLEJIDAD COMPUTACIONAL

La teoría de la complejidad computacional es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad inherente y en la relación entre las clases de complejidad.

Un problema se cataloga como inherentemente difícil si su solución requiere una cantidad significativa de recursos computacionales.

Uno de los fines de la teoría de la complejidad computacional es determinar los límites prácticos de qué es lo que se puede hacer con una máquina y qué no.

Clase de complejidad	Modelo de cómputo	Restricción de recurso
DTIME($f(n)$)	Máquina de Turing determinista	Tiempo $f(n)$
P	Máquina de Turing determinista	Tiempo poly(n)
EXPTIME	Máquina de Turing determinista	Tiempo $2^{poly(n)}$
NTIME($f(n)$)	Máquina de Turing no determinista	Tiempo $f(n)$
NP	Máquina de Turing no determinista	Tiempo poly(n)
NEXPTIME	Máquina de Turing no determinista	Tiempo $2^{poly(n)}$
DSPACE($f(n)$)	Máquina de Turing determinista	Espacio $f(n)$
L	Máquina de Turing determinista	Espacio O(log n)
PSPACE	Máquina de Turing determinista	Espacio poly(n)
EXPSPACE	Máquina de Turing determinista	Espacio $2^{poly(n)}$
NSPACE($f(n)$)	Máquina de Turing no determinista	Espacio $f(n)$
NL	Máquina de Turing no determinista	Espacio O(log n)
NPSPACE	Máquina de Turing no determinista	Espacio poly(n)
NEXPSPACE	Máquina de Turing no determinista	Espacio $2^{poly(n)}$

Fig. 4. Clases de complejidad importantes.

La relación entre las clases de complejidad P y NP es una cuestión que todavía no se ha podido resolver.

La clase P consta de todos aquellos problemas de decisión que pueden ser resueltos en una máquina determinista secuencial en un periodo de tiempo polinomial en proporción a los datos de la entrada. La clase P contiene la mayoría de problemas naturales, algoritmos de programación lineal o funciones simples.

La clase NP consta de todos aquellos problemas de decisión cuyas soluciones pueden ser halladas en una máquina no determinista. Un ejemplo de problema de la clase NP es el del camino máximo de un grafo.

Las reducciones en tiempo polinomial nos dotan de elementos para probar si un problema es tan difícil como otro, con una diferencia de un factor polinomial. Estas son esenciales para definir los problemas NP-completos, además de ayudar a comprender los mismos.

La clase de los problemas NP-completos contiene a los problemas más difíciles de la clase NP, es decir, son aquellos que están más alejados de estar en la clase P. Son NP-completos los problemas de la satisfacibilidad booleana (SAT) y la resolución de un laberinto.

Esta clase tiene la propiedad de que si algún problema NP-completo puede ser resuelto en tiempo polinomial, entonces todo problema NP tiene una solución en tiempo polinomial. [8]

5 – COMPUTACIÓN CUÁNTICA

5.1 – INTRODUCCIÓN

Muchos problemas de las ciencias son imposibles de resolver en los ordenadores tradicionales, no por ser irresolubles, sino por la enorme cantidad de recursos necesarios para su resolución.

La *computación cuántica* se basa en el desarrollo de un conjunto de nuevos algoritmos capaces de resolver todos esos problemas en los ordenadores tradicionales.

Hoy en día existen dos clases de *algoritmos cuánticos* que se utilizan en la resolución de estos problemas. El primero está basado en la transformada de Fourier cuántica de Shor, que se compone de varios algoritmos para factorizar y hallar logaritmos discretos que son mucho más rápidos que los usados en los ordenadores convencionales. El segundo está basado en el algoritmo de búsqueda cuántica de Grover, que provee de una gran aceleración sobre los mejores algoritmos tradicionales.

Se han desarrollado muy pocos *algoritmos cuánticos* ya que el diseño de algoritmos óptimos es complejo para problemas simples y buscar buenos *algoritmos cuánticos* supone tener que superar a los algoritmos clásicos. Otro motivo es que nuestra intuición está mejor adaptada a la mecánica clásica que a la *mecánica cuántica*. [1]

5.2 – BASES DE LA COMPUTACIÓN CUÁNTICA

5.2.1 – BITS CUÁNTICOS

El bit es el concepto fundamental de la computación clásica. La *computación cuántica* se construye en torno a un concepto análogo, el *bit cuántico* o *qubit*. Un *qubit* es un objeto matemático que tiene un estado. Los posibles estados de un

qubit son $|0\rangle$ y $|1\rangle$, que corresponden con los estados 0 y 1 del bit. **La diferencia entre los bits y los *qubits* es que los *qubits* pueden encontrarse en un estado distinto de $|\mathbf{0}\rangle$ o $|\mathbf{1}\rangle$, lo que permite formar combinaciones lineales de estados, denominadas superposiciones**

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{15}$$

Los números ***α*** y ***β*** son números complejos, aunque para muchos propósitos podemos considerarlos números reales. Dicho de otro modo, el estado de un *qubit* es un vector de dos dimensiones en el espacio complejo. Los estados especiales $|0\rangle$ y $|1\rangle$ se conocen como los estados computacionales base y forman la base ortonormal para este espacio de vectores.

Para determinar en qué estado se encuentra un bit podemos examinarlo, pero no podemos hacer lo mismo con un *qubit*. Cuando medimos un *qubit* podemos tener el resultado *0* con probabilidad $|\alpha|^2$ o el resultado *1* con probabilidad $|\beta|^2$. Esto hace que

$$|\alpha|^2 + |\beta|^2 = 1 \tag{16}$$

Con la ecuación (16) podemos reescribir la ecuación (15) como

$$|\Psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sen\frac{\theta}{2}|1\rangle \tag{17}$$

Los números ***θ*** y ***ϕ*** definen un punto de la esfera unitaria tridimensional conocida como la esfera de Bloch. [1]

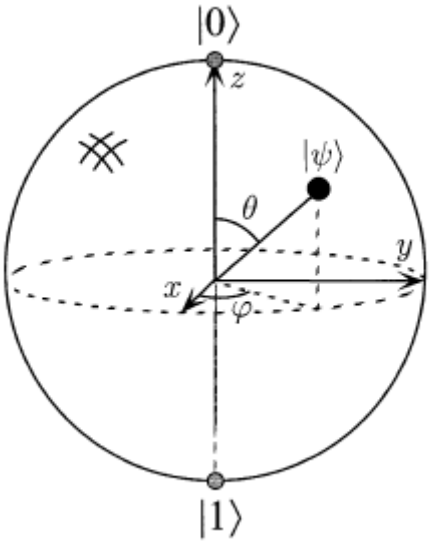

Fig. 5. Representación de un qubit en la esfera de Bloch.

5.2.2 – QUBITS MÚLTIPLES

Supongamos que tenemos dos *qubits*. Si fueran dos bits clásicos, entonces tendríamos cuatro estados posibles: 00, 01, 10 y 11. Acorde a esto, un sistema de dos *qubits* tiene cuatro estados computacionales base denominados $|00\rangle$, $|01\rangle$, $|10\rangle$ y $|11\rangle$. **Un par de *qubits* pueden existir en superposición de estos cuatro estados, de tal modo que el estado cuántico de dos *qubits* involucra asociar un coeficiente complejo a cada estado computacional base**, de manera que el vector de estado que describe esos dos *qubits* es [1]

$$|\Psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \qquad (18)$$

5.2.3 – PUERTAS LÓGICAS DE QUBITS SIMPLES

Los circuitos de un ordenador clásico se componen de cables y puertas lógicas. Los cables se usan para transportar información por el circuito y las puertas lógicas realizan cambios en la información, convirtiéndola de una forma a otra.

Para desarrollar herramientas computacionales cuánticas es necesario poder realizar operaciones sobre los *qubits*, para lo que se utilizarán las *puertas lógicas cuánticas*. **Estas puertas cumplen la función equivalente a la de los operadores booleanos unitarios de la computación clásica. La combinación de estas puertas permite el desarrollo en microcódigo de los *algoritmos cuánticos*, para ser trasladados a hardware si se dan las condiciones suficientes.**

Las puertas lógicas deben cumplir las normas de los *qubits* y por lo tanto deben ser operadores unitarios 2x2. Las puertas lógicas más importantes son las siguientes [1]

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (19)$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}; S = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix}; T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \quad (20)$$

La Fig. 6 muestra los diagramas básicos de estas puertas.

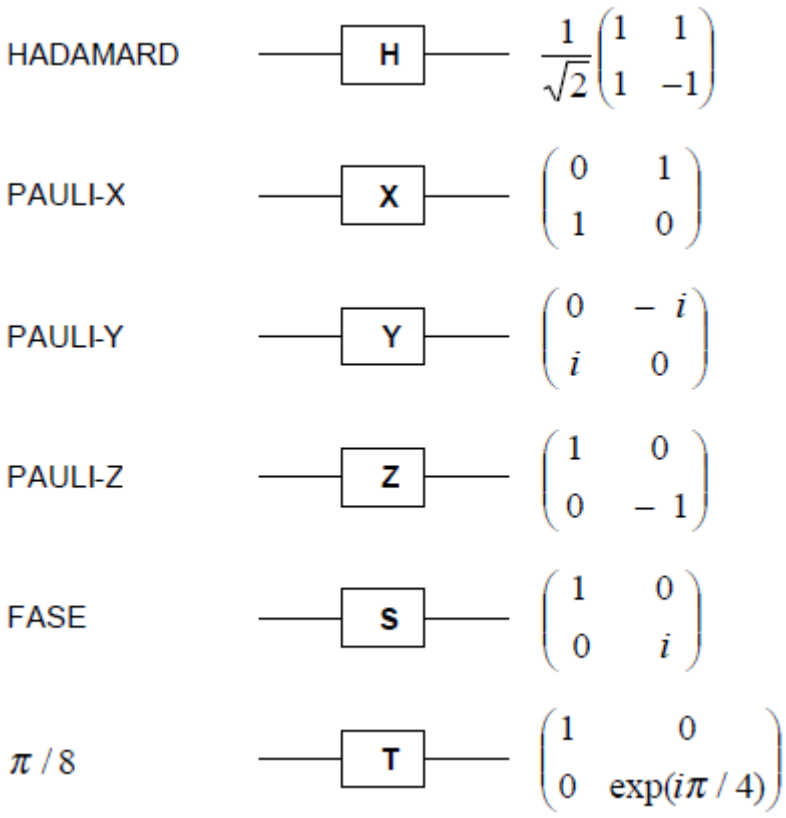

Fig. 6. Puertas lógicas de qubits simples.

5.2.4 – PUERTAS LÓGICAS DE QUBITS MÚLTIPLES

Una de las operaciones más útiles tanto en la computación clásica como en la *computación cuántica* es la bifurcación condicional del tipo "Si *A* es cierto entonces ejecute *B*". Estas bifurcaciones condicionales operan sobre dos *qubits* y por lo tanto serán matrices 4x4. El prototipo de estas operaciones es la puerta CNOT representada en la Fig. 7.

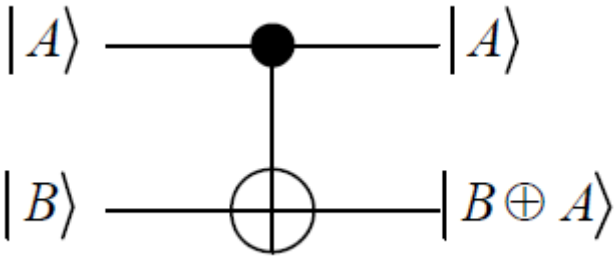

Fig. 7. Puerta CNOT.

La línea superior es el *qubit* de control y la inferior es el *qubit* controlado. La operación es la siguiente

$$|A\rangle|B\rangle \rightarrow |A\rangle|B \oplus A\rangle \tag{21}$$

donde si A es $|1\rangle$ se realiza la operación NOT sobre B y si A es $|0\rangle$ B no se modifica.

La puerta CNOT clásica permite duplicar bits, pero sin embargo la CNOT cuántica no puede duplicar *qubits*, lo que demuestra que **los estados cuánticos no pueden ser clonados**. La analogía clásica de una puerta CNOT es una puerta XOR reversible.

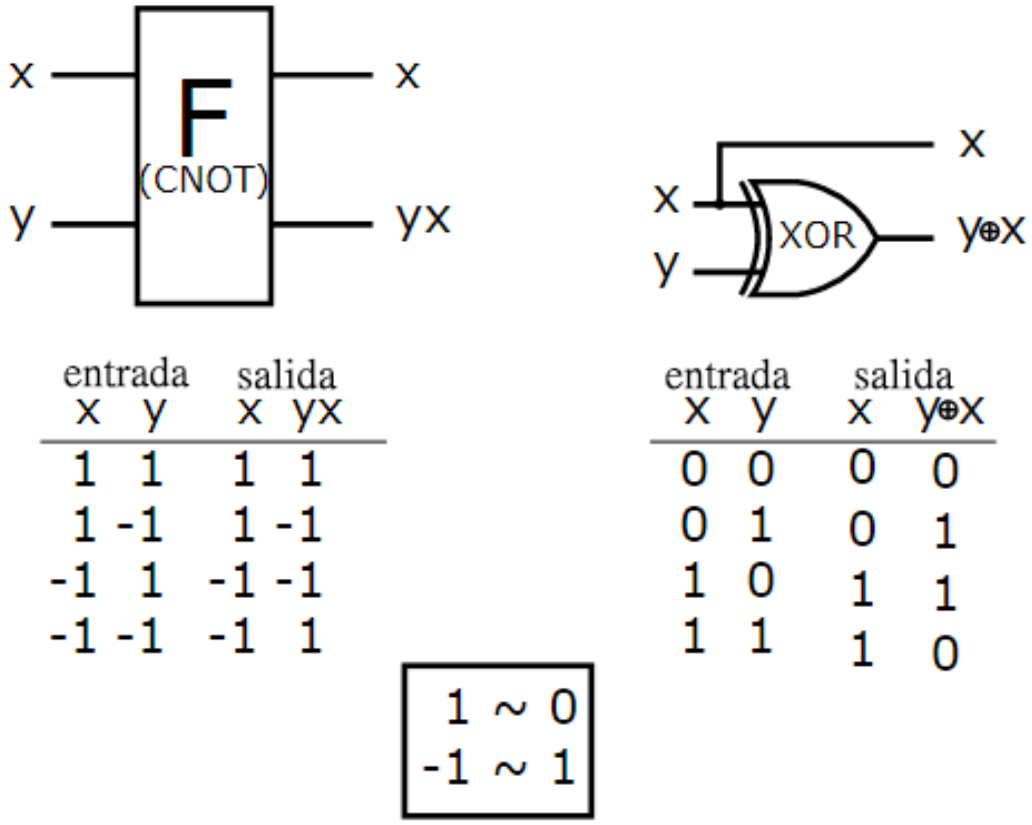

Fig. 8. Analogía entre la CNOT cuántica y la CNOT clásica.

Con las puertas Hadamard (H), Fase (S), CNOT y π/8 se puede construir cualquier circuito cuántico basado en operaciones unitarias. Todos los circuitos binarios o de orden superior se pueden transformar en un circuito unitario y estas puertas lógicas pueden construir cualquier *computador cuántico*.

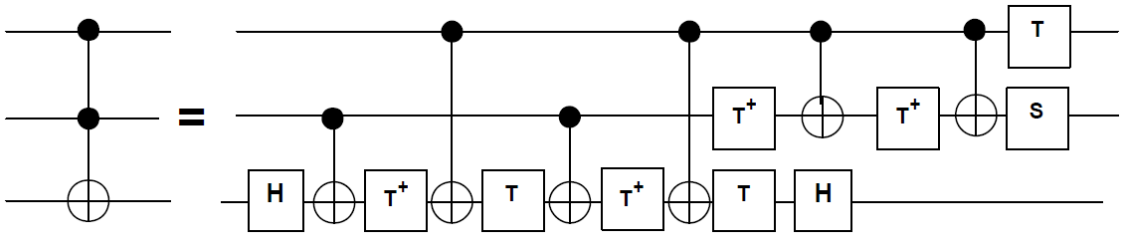

Fig. 9. Descomposición secuencial de la puerta Toffoli en operadores unarios elementales. [9]

Por último, tenemos la puerta de medición, que transforma un *qubit* en un bit clásico.

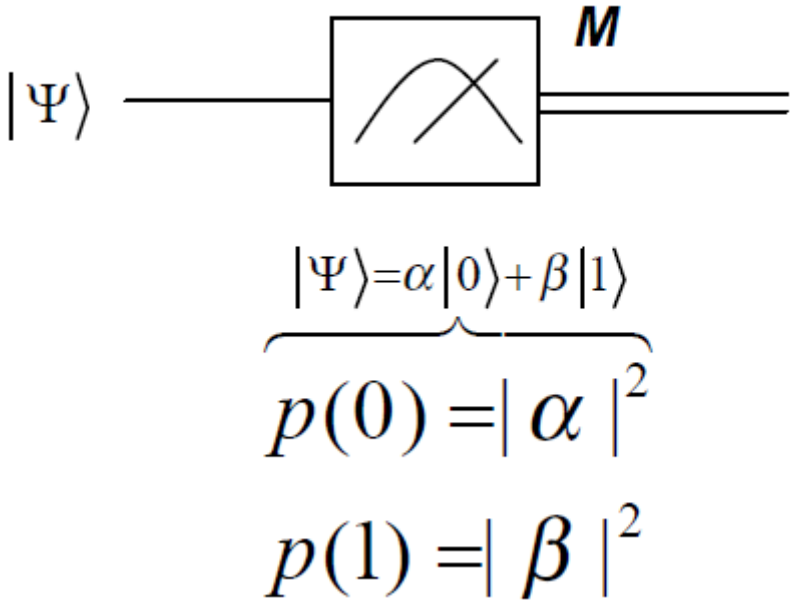

Fig. 10. Puerta de medición.

La Fig. 10 representa una puerta de medición y las líneas dobles se usan para representar los bits clásicos. [1]

5.3 – CRITERIOS DE LA COMPUTACIÓN CUÁNTICA

La *computación cuántica* fue propuesta por Richard Feynman en 1982 como un medio eficiente de simular *sistemas cuánticos*. Desde entonces se han realizado muchas propuestas para construir un *ordenador cuántico* con distintos grados de éxito en los retos de construir *dispositivos cuánticos*. Algunas de estas propuestas incluyen el uso de *qubits superconductores*, *trampas de iones* o *resonancias magnéticas nucleares de estado sólido y líquido*, todas ellas tienes aspectos muy interesantes, pero surgen problemas a la hora de una implementación práctica. El criterio de DiVincenzo es una lista de condiciones necesarias para la construcción de un *ordenador cuántico*.

Se trata de cinco más dos condiciones que un diseño experimental debe satisfacer para implementar *algoritmos cuánticos*. Las dos condiciones

adicionales son necesarias para implementar la *comunicación cuántica*. Este criterio de DiVincenzo se puede trasladar a la computación clásica y demostrar que se satisface.

- **Sistema físico escalable con *qubits* bien caracterizados.** La mayoría de modelos de *computación cuántica* requieren el uso de *qubits* para la computación. Esto puede ser difícil de implementar físicamente, especialmente en la transición entre niveles atómicos. Independientemente del sistema que escojamos, se requiere el sistema permanezca casi siempre en el subespacio de los estados computacionales base, de tal manera que lo podamos considerar un *qubit* bien caracterizado.

- **Capacidad de inicializar el estado de los *qubits* con un marcador de referencia.** Todos los modelos de *computación cuántica* se basan en la realización de operaciones en el estado del *qubit* y finalmente medir el resultado, un procedimiento que depende del estado inicial del sistema. Particularmente, la naturaleza única de la *mecánica cuántica* hace que la inicialización de los *qubits* sea extremadamente importante. En muchos casos la forma de inicializar un estado es dejar que el sistema se encuentre en su estado fundamental y después iniciar la computación.

- **Tiempos de decodificación lo suficientemente largos.** La semejanza a un sistema clásico de un gran *sistema cuántico* es debida al incremento de la decodificación necesaria. La escala de tiempos asociada con esta pérdida de *comportamiento cuántico* se vuelve importante cuando se construyen grandes sistemas de *computación cuántica*.

- **Un conjunto universal de *puertas lógicas cuánticas*.** Los algoritmos de computación que se pueden implementar están restringidos por el número de puertas que se pueden implementar. En el caso de la *computación cuántica* ya hemos podido observar que podemos construir cualquier *circuito cuántico* con un pequeño conjunto de *puertas lógicas cuánticas*.

- **La capacidad de medir *qubits* específicos.** Para cualquier proceso aplicado al estado cuántico de qubits la medición final es de fundamental importancia cuando se realizan cambios en la información.

- **La capacidad de interconventir *qubits estacionarios* y *qubits flotantes* y la capacidad de llevar correctamente *qubits flotantes* a lugares específicos.** Estas dos condiciones son necesarias únicamente cuando consideramos los protocolos de *comunicación cuántica*. [10]

5.4 – MÁQUINAS UNIVERSALES CUÁNTICAS

5.4.1 – INTRODUCCIÓN

Todos los modelos de computación existentes son clásicos. Esto significa que una especificación completa de su estado en cualquier instante es equivalente a la especificación de un conjunto de números, los cuales son medibles. De acuerdo con la *teoría cuántica* no existen sistemas físicos con esta propiedad, pero sabemos que la física clásica ha quedado anticuada.

En 1982 Benioff construyó un modelo de computación con cinemáticas y *dinámicas cuánticas*, pero tras realizar las comprobaciones se descubrió que el modelo era clásico.

También en 1982 Feynman se acercó a la construcción de un verdadero *ordenador cuántico* con su "simulador cuántico universal". Aunque se podía simular cualquier sistema con un número finito de dimensiones del espacio de estados no se trata de una máquina de computación como tal.

Para resolver la necesidad, Deutsch desarrolló un *modelo computacional cuántico* y una *máquina universal cuántica*. [11]

5.4.2 – MÁQUINA UNIVERSAL CUÁNTICA

Como en una máquina de Turing, el modelo de una *máquina cuántica* está formado por dos componentes, un procesador finito y una memoria infinita, de la cual solo una pequeña porción se usará. La computación se realiza en pasos de una duración fija y durante cada paso solo el procesador y una parte de la memoria interactúan, el resto de la memoria permanece estática.

El procesador contiene dos estados *M* observables

$$\{\hat{n}_i\} \quad (i \in \mathbb{Z}_M) \tag{22}$$

donde $\mathbb{Z}_M$ es el conjunto de enteros de *0* a *M-1*.

La memoria consiste en una secuencia infinita de dos estados observables

$$\{\hat{m}_i\} \quad (i \in \mathbb{Z}) \tag{23}$$

Esto se corresponde a la cinta de memoria infinita de una máquina de Turing.

Llamaremos a los $\{\hat{n}_i\}$ como $\hat{n}$ y a los $\{\hat{m}_i\}$ como $\hat{m}$. Respecto a la posición de la cinta existe una observable $\hat{x}$ que puede tener cualquier valor entero $\mathbb{Z}$. La observable $\hat{x}$ corresponde a la dirección actual de la cinta. Por lo tanto, el estado de la máquina es un vector unitario en el espacio de Hilbert desplegado por los autovectores simultáneos

$$|x; n; m\rangle \equiv |x; n_0, n_1, \dots, n_{M-1}; \dots, m_{-1}, m_0, m_1, \dots\rangle \tag{24}$$

de $\hat{x}$, $\hat{n}$ y $\hat{m}$, que se nombran como sus correspondientes autovalores *x*, *n* y *m*.

Esta ecuación (10) es la definición de los estados computacionales base. Es conveniente tomar el espectro de nuestra observable de dos estados para que sea $\mathbb{Z}_2$, es decir, {0, 1} y no en la base física tradicional $\left\{-\frac{1}{2}, +\frac{1}{2}\right\}$ para su interpretación en *qubits*.

La dinámica de la máquina se resume por un operador unitario constante *U* en el espacio de Hilbert. Este operador especifica la evolución de cualquier estado $|\Psi(t)\rangle$ ϵ *H* durante un paso computacional, mientras que para *n* pasos tendríamos

$$|\Psi(nT)\rangle = U^n|\Psi(0)\rangle \quad (n \in \mathbb{Z}_+) \tag{25}$$

donde $U^+U=UU^+=I$. No es necesario especificar el estado para valores negativos de tiempo ya que la computación comienza en *t=0*. En ese tiempo inicial $\hat{x}$ y $\hat{n}$ se inicializan con valor cero y el estado de un número finito de $\hat{m}$ se prepara como el programa y la entrada y el resto se deja a cero. Por lo tanto [9]

$$\begin{cases} |\Psi(0)\rangle = \sum_m \lambda_m |0;0;m\rangle \\ \sum_m |\lambda_m|^2 = 1 \end{cases} \tag{26}$$

donde solo un número finito de los λ_m son no nulos y λ_m desaparece cuando un número infinito de los *m* sean no nulos. Para que la máquina opere por medios finitos, los elementos de la matriz *U* toman la siguiente forma

$$\langle x';n';m'^{|U|}x;n;m\rangle = \\ [\delta_{x'}^{x+1}U^+(n',m'_x|n,m_x) + \delta_{x'}^{x-1}U^-(n',m'_x|n,m_x)]\prod_{y\neq x}\delta_{m_y}^{m_y} \tag{27}$$

donde el producto asegura que sólo un *qubit* sea transformado en cada paso computacional. Los otros dos términos de Kronecker aseguran que durante cada paso la posición de la cinta no pueda cambiar en más de una unidad, mientras que las funciones $U^{\pm}(n^*,m^*_x|n,m_x)$ representan un proceso dinámico que depende de las observables locales $\hat{n}$ y $\widehat{m_x}$, que son arbitrarias excepto que sean unitarias. Cada elección de esas funciones define una máquina cuántica diferente.

Las máquinas de Turing se detienen, señalizando así el fin de la computación. En el caso de las máquinas cuánticas esa condición se alcanzaría cuando dos

estados globales consecutivos fuesen idénticos y así el sistema llegaría al equilibrio. Sin embargo, la ecuación (25) asegura que después de una transformación unitaria no trivial nunca se consigue que el sistema final sea igual al sistema inicial.

Ninguna máquina cuática debe ser observada antes de que la computación haya finalizado ya que esto alteraría su estado. Por este motivo, las máquinas cuánticas necesitan señalar que han finalizado la computación. Uno de los *qubits* (n_0) de la cinta se reservará con este propósito. Cada programa válido fuerza $n_0 = 1$ cuando termina, sin que se interactúe con n_0 durante la ejecución del programa. Por lo tanto, el *qubit* n_0 podrá ser medido periódicamente para ver si se llegó a la detención. Un *programa cuántico* es válido si la esperanza de su tiempo de procesamiento es finita.

Dado que, *U* es unitaria, la dinámica de la máquina, como la de cualquier sistema cerrado, es reversible. Se pueden obtener *máquinas cuánticas* equivalentes a cualquier máquina clásica reversible de Turing usando

$$U^{\pm}(n^*, m^*|n, m) = \frac{1}{2}\delta_{n^*}^{A(n,m)}\delta_{m^*}^{B(n,m)}[1 \pm C(n, m)] \tag{28}$$

donde A, B y C son funciones con rangos $(\mathbb{Z}_2)^M$, $\mathbb{Z}_2$ y *(-1, +1)* respectivamente. En otras pablaras, las máquinas de Turing son aquellas *máquinas cuánticas* cuyas dinámicas aseguran que permanecen en los estados computacionales base al finalizar cada paso si se asegura que partieron de uno. La condición necesaria y suficiente de unitariedad es que el siguiente mapa sea biyectivo

$$\{(n, m)\} \leftrightarrow \{\big(A(n, m), B(n, m), C(n, m)\big)\} \tag{29}$$

Dado que las funciones A, B y C son arbitrarias en todo sentido no específicamente limitado, entonces deben existir opciones que hagan la *máquina cuántica* equivalente a la máquina universal de Turing. [11]

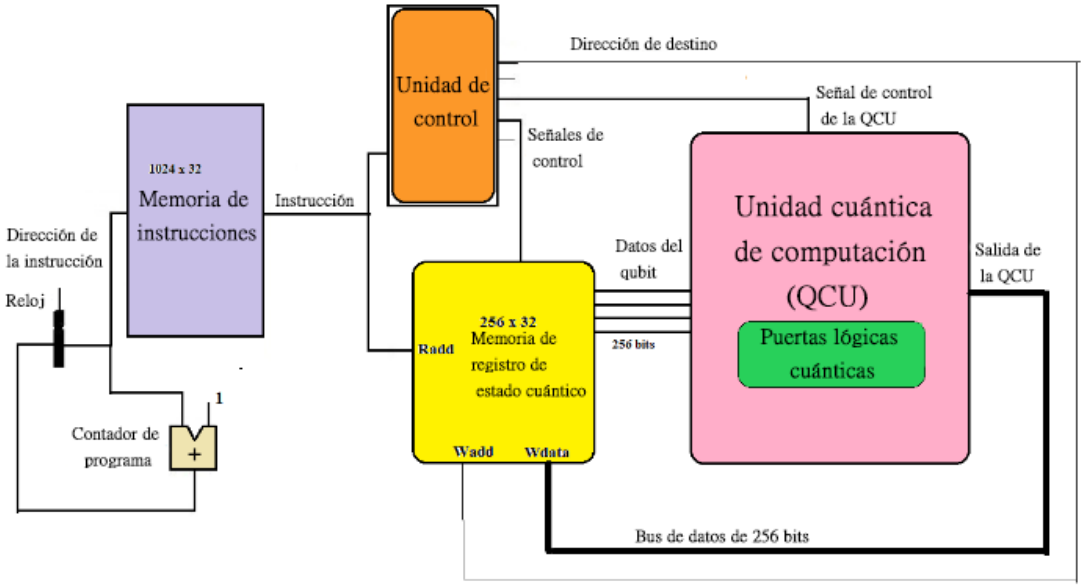

Fig. 11. Ejemplo de procesador cuántico.

5.5 – SOLUCIÓN DE PROBLEMAS NP-COMPLETOS

Encontrar un método para resolver eficientemente problemas NP-completos es una de las grandes dificultades de la teoría de la computación. Ahora se estudiará un método eficiente de resolver problemas NP-completos usando la *máquina universal cuántica*.

Siendo *C* una configuración específica, podemos observar la existencia de *C* en una superposición con certeza en tiempo polinómico en la entrada de *C*, si *C* existe en la superposición.

A continuación, se muestra un programa *P* que *Q* (UQTM) simula para resolver el problema SAT en tiempo polinómico. Se considera la satisfacción de una fórmula de *m*-variables lógicas $f(x_1, x_2, \ldots, x_m)$, donde x_i, $i = 1, 2, \ldots, m$ son variables booleanas.

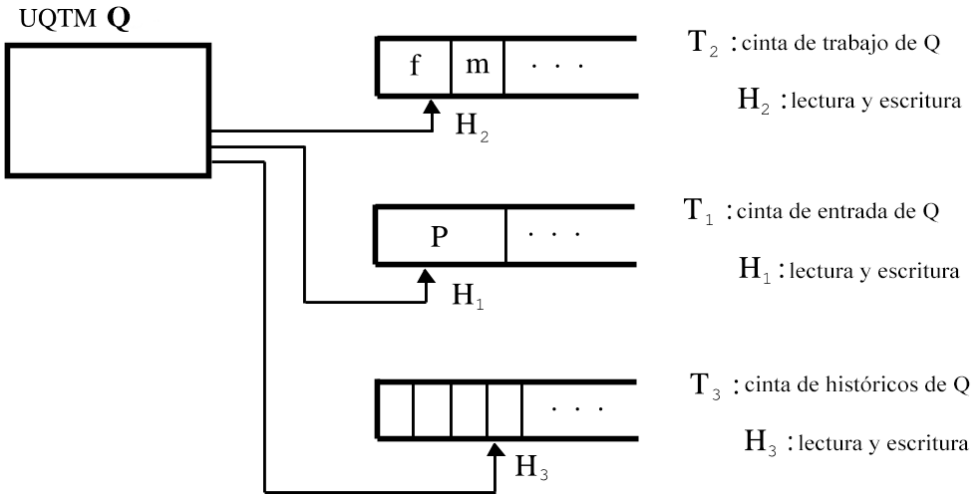

Fig. 12. La UQTM *Q*.

Q escribe una asignación a las variables de *f* junto con el correspondiente valor de *f* en la cinta T_2. Se muestra el programa simulado por *U* en la Fig. 13.

```
begin
%%% Preparación de las asignaciones
1       for i = 1 to m do
2           V(4, i)
        od ;
%%% Cálculo de los valores de f
3       x_{m+1} := f (x_1, ..., x_m) ;
%%% Preparación para la observación
4       for j = m to 1 do
5           V(0, j)
        od
end.
```

Fig. 13. Programa simulado por la UQTM.

El comportamiento de *Q* es el siguiente:

1. **Configuración inicial**: Las descripciones de la fórmula lógica *f* y el número *m* de variables vienen dadas como entrada en la cinta de trabajo T_2, y el programa *P* que *Q* simula se da en la cinta de entrada T_1.
2. **Hacer una superposición de todas las asignaciones**: Existen 2^m asignaciones diferentes para *m* variables en *f*. Inicialmente *Q* realiza una superposición de configuraciones correspondiente a todas las asignaciones. *Q* realizará esto aplicando

$$V_4 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \tag{30}$$

   a cada bit correspondiente a las *m* variables. *Q* puede ejecutar la transformación mediante V_4 en un único paso. Cuando *Q* completa la ejecución del bucle for de la Fig. 11. el histórico $Z_1, Z_2, \ldots, Z_m$ se escribe en T_3 en cada configuración contenida en la superposición obtenida de las configuraciones de *Q*. Aquí, Z_{i-1} $(2 \leq i \leq m)$ es el histórico de las transiciones de *Q* desde la configuración inmediatamente posterior a la ejecución de *V(4, i-1)* hasta la configuración inmediatamente posterior a la ejecución de *V(4, i)*.

3. **Cálculo de los valores de *f***: Para ejecutar la tercera línea del programa, *Q* simula el programa de una cinta M_f para calcular los valores de *f*. *Q* simula M_f, siendo T_2 la cinta de M_f. Al final *f* y *m* se escriben en T_2, seguidos de una asignación $[a_1, \ldots, a_m]$ $(a_i \in \{0, 1\}, 1 \leq i \leq m)$ para $x_1, \ldots, x_m$.

Acción	T_2	T_1	T_3
Estado inicial	$f, m, [a_1, \ldots, a_m, 0]$	P	$Z_1 Z_2 \cdots Z_m$
Cálculos	$f, m, [a_1, \ldots, a_m, a_{m+1}]$	P	$Z_1 Z_2 \cdots Z_m$, Histórico
Copia reversible $(T_2 \rightarrow T_1)$	$f, m, [a_1, \ldots, a_m, a_{m+1}]$	P, a_{m+1}	$Z_1 Z_2 \cdots Z_m$, Histórico
Borrado	$f, m, [a_1, \ldots, a_m, 0]$	P, a_{m+1}	$Z_1 Z_2 \cdots Z_m$
Movimiento reversible $(T_1 \rightarrow T_2)$	$f, m, [a_1, \ldots, a_m, a_{m+1}]$	P	$Z_1 Z_2 \cdots Z_m$

Fig. 14. Cálculo de los valores de *f*.

4. **Preparación de la observación**: *Q* realiza la transformación inversa a la realizada en el paso 2, aplicando

$$V_0 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \tag{31}$$

a los bits correspondientes a las *m* variables, desde $[x_m]$ hasta $[x_1]$ en ese orden. Ya que *Q* puede realizar la transformación correspondiente a V_0 en un único paso, se puede ejecutar el bucle for de la cuarta línea en *m* pasos.
Cuando se finaliza la transformación, los históricos $Z_1, Z_2, \ldots, Z_m$ y $W_m, \ldots, W_2, W_1$ se escriben en T_3 en todas las configuraciones existentes de la superposición obtenida de las configuraciones de *Q*. Aquí, W_m es el histórico de la transición desde la configuración inmediatamente posterior a la tercera línea hasta la configuración inmediatamente anterior a la ejecución de *V(0, m)* y $W_i(1 \leq i \leq m)$ es el histórico de la transición desde la configuración inmediatamente posterior a la ejecución de *V(0, i+1)* hasta la configuración inmediatamente anterior a la ejecución de *V(0, i)*.

Cuando *Q* completa los cálculos anteriores podemos observar si existe una configuración c_0 en la superposición de configuraciones de *Q*. Si c_0 existe en la superposición, podemos afirmar que *f* se satisface y si no existe, *f* no se satisface.

A continuación, se muestran los cambios de la superposición de las configuraciones de *Q* en el caso de una función f tal que *f(0, 0) = f(1, 1) = 0* y *f(0, 1) = f(1, 0) = 1*.

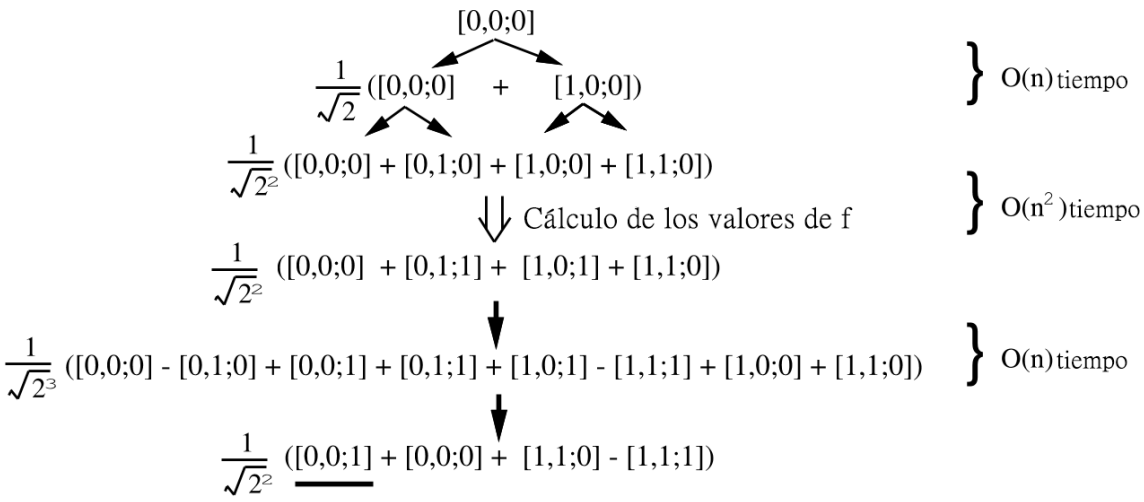

Fig. 15. Resolución del SAT.

Como podemos observar, la resolución de problemas NP-completos es posible usando la UQTM y para la resolución de otros problemas NP-completos habría que adaptar este algoritmo de resolución al problema propuesto. [12]

6 – MEMCOMPUTACIÓN

6.1 – INTRODUCCIÓN

Hoy en día, en los ordenadores convencionales, el almacenamiento y el proceso de la información ocurre en distintas regiones físicas de espacio. La información se almacena en memorias volátiles (RAM) y no volátiles (discos duros) y después es procesada secuencialmente por una unidad central de proceso (CPU). Esto no solo se traduce en limitaciones de espacio, sino que produce retardos indeseados en la recuperación y el proceso de información importante, ya que este modo de operación requiere una transferencia de gran volumen de información entre la CPU y las unidades de memoria y viceversa, además también impone límites en el rendimiento y la escalabilidad de la arquitectura. Por ejemplo, en la arquitectura tradicional de von Neumann el ratio de transferencia de los datos entre la CPU y las unidades de memoria es el factor que limita la velocidad de computación.

Una posibilidad para reducir parcialmente este problema es el empleo de la computación en paralelo, en la que la ejecución del programa se distribuye a través de múltiples núcleos de procesamiento que acceden a una memoria local con un ratio más rápido que a la memoria no local de los otros procesadores. Sin embargo, aunque se usan múltiples núcleos de computación en las CPU modernas, la escalabilidad en una estación de trabajo se consigue únicamente con unidades especializadas.

Por lo tanto, un cambio no incremental en el rendimiento de la computación requiere un cambio de paradigma de la arquitectura tradicional de von Neumann a nuevos y eficientes esquemas de computación masiva en paralelo, probablemente basados en dispositivos electrónicos no convencionales. La *computación cuántica*, de la que ya hemos hablado anteriormente, ha sido considerada una posible solución desde su creación. Sin embargo, no se ha podido fabricar todavía un ordenador cuántico que cumpla con las expectativas y necesidades para competir con un ordenador clásico.

Entonces, ¿se podría **formular una computación alternativa que fuera capaz de trabajar con información masiva en paralelo y cuyas unidades de almacenamiento y proceso fueran físicamente las mismas**? Nuestro cerebro parece cumplir con estas condiciones. Aunque no se conoce la topología del cerebro humano completamente, sabemos que las neuronas y sus conexiones almacenan y procesan información simultáneamente, y su funcionamiento es colectivo y adaptativo. Por otra parte, el funcionamiento del cerebro es estable con respecto al fallo de algunas neuronas. Por lo tanto, nuestro cerebro cuenta con un mecanismo interno con la capacidad de evitar las conexiones rotas con caminos alternativos.

Para reproducir estas características en circuitos electrónicos, necesitaríamos elementos que se adaptaran a la señal entrante y retuvieran información bajo demanda. Las arquitecturas tradicionales basadas en transistores realmente cumplirían estos requisitos, pero los transistores son dispositivos activos de tres terminales y su funcionamiento se produce a costa de un consumo de energía relativamente alto y una baja densidad. Aunque es poco probable que los elementos activos sean eliminados de la electrónica, sería deseable mantenerlos bajo mínimos y, en su lugar, dejar el almacenamiento y el procesamiento de la información a los elementos pasivos, preferiblemente con dimensiones de escala nanométrica.

Por todo esto, el paradigma de la *memcomputación* utilizará elementos de un circuito con memoria (*memelementos*). [13]

6.2 – MEMELEMENTOS

6.2.1 – INTRODUCCIÓN

Los memelementos son elementos de un circuito electrónico capaces de almacenar información sin la necesidad de una fuente de energía, lo que permite la computación y el almacenamiento de baja potencia. Además, si esa información abarca un rango continuo de valores, la computación analógica

podría reemplazar a la computación digital actual. El origen de este concepto se basa en el funcionamiento del cerebro humano y en otros mecanismos de organismos vivos, de tal modo que estos elementos de un circuito nos podrían ayudar a entender el comportamiento adaptativo y espontáneo e incluso el aprendizaje.

Uno de estos elementos de un circuito es el *memresistor*, que fue postulado por L.O. Chua en 1971 mediante el análisis de relaciones matemáticas entre pares de variables de un circuito fundamental. El *memresistor* está definido por una relación entre la carga y el flujo, definida matemáticamente como la integral en el tiempo de la tensión, que no necesita tener una interpretación de flujo magnético. Esta relación se puede generalizar para incluir cualquier clase de dispositivo de dos terminales (*sistema memresistivo*) cuya resistencia depende del estado interno del sistema.

Muchos sistemas pertenecen a esta clase, incluyendo el termistor (cuyo estado interno depende de la temperatura), moléculas cuya resistencia cambia según su configuración atómica o dispositivos cuya resistencia varía según su configuración de espín. El *comportamiento memresistivo* y el almacenamiento en memoria han sido probados en películas finas de TiO_2 de estado sólido, donde el cambio en la resistencia se realiza mediante el movimiento iónico de las vacantes de oxígeno activadas por el flujo de corriente. Además, el *comportamiento memresistivo* se ha demostrado en películas finas de VO_2, donde **el mecanismo de memoria está relacionado con la transición de aislante a conductor** de estas estructuras. Finalmente, el *comportamiento memresistivo* ha sido identificado por M. Di Ventra y Yu. V. Pershin como un posible mecanismo en el comportamiento adaptativo de organismos unicelulares.

Estos ejemplos demuestran la naturaleza omnipresente de los *sistemas memresistivos*. De hecho, muchos de los ejemplos anteriores se refieren a sistemas nanoscópicos, cuya resistencia dependerá de su estado y de su historia dinámica, al menos dentro de muy pequeñas escalas de tiempo dictadas por las variables de estado fundamentales que controlan su funcionamiento.

Este concepto de dispositivo de memoria no se limita únicamente a las resistencias, sino que de hecho se puede generalizar a sistemas capacitivos e inductivos. En general, si *x* indica un conjunto de *n* variables de estado que describen el estado interno del sistema, *u(t)* e *y(t)* son dos variables complementarias constitutivas (ejemplos: corriente, carga, tensión o flujo) que denotan la entrada y la salida del sistema, y *g* es una respuesta generalizada, podemos definir una clase general de dispositivos de memoria de orden *n* y *u*-controlados como los descritos por las siguientes relaciones

$$y(t) = g(x, u, t) \cdot u(t) \tag{32}$$

$$\dot{x} = f(x, u, t) \tag{33}$$

donde *f* es una función vectorial *n*-dimensional continua, y asumimos por razones físicas que, dado un estado inicial $u(t = t_0)$ en el instante t_0, la ecuación (33) tiene solución única.

Los sistemas *memcapacitivos* y *meminductivos* son casos especiales de las ecuaciones (32) y (33), donde las dos variables que los definen son carga y tensión para la *memcapacitancia*, y corriente y flujo para la *meminductancia*. [14]

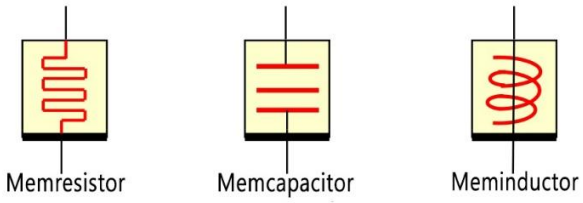

Fig. 16. Símbolos de los diferentes dispositivos: *memresistor*, *memcapacitor* y *meminductor*.

6.2.2 – SISTEMAS MEMRESISTIVOS

Primero se va a introducir la noción de sistemas resistivos con memoria con un ejemplo analítico del *comportamiento memresistivo*.

Desde las ecuaciones (32) y (33) se puede describir un *sistema memresistivo* controlado por corriente de orden *n* mediante las ecuaciones

$$V_M(t) = R(x, I, t) \cdot I(t) \quad (34)$$

$$\dot{x} = f(x, I, t) \quad (35)$$

siendo *x* un vector de estados que representa *n* variables de estado internas, $V_M(t)$ e $I(t)$ indican la tensión y la corriente a través del dispositivo, y *R* es un escalar denominado *memresistencia*, que se mide en Ohmios. La ecuación de un *memresistor* controlado por carga es un caso particular de las ecuaciones (34) y (35), donde *R* depende solo de la carga

$$V_M = R(q(t)) \cdot I \quad (36)$$

estando la carga relacionada con la corriente mediante la derivada: $I=dq/dt$.

También podemos definir un *sistema memresistivo* controlado por tensión de orden *n* mediante las ecuaciones

$$I(t) = G(x, V_M, t) \cdot V_M(t) \quad (37)$$

$$\dot{x} = f(x, V_M, t) \quad (38)$$

donde *G* es la *memductancia*.

Algunas de las propiedades más importantes de los *sistemas memresistivos* se describen a continuación. Considerando dispositivos con $R(x, I, t) > 0$ en la ecuación (34), se ha probado que estos dispositivos son pasivos. Siguiendo esta inecuación se demuestra el hecho de que un *sistema memresistivo* no puede almacenar energía, como un capacitor o un inductor. Como manifestación de estas características podemos deducir de la ecuación (34) que $V_M = 0$ cuando $I = 0$ (y viceversa). Además, para una entrada periódica de corriente, un *sistema memresistivo* muestra un lazo de histéresis pellizcado.

Un *sistema memresistivo* se comporta como un resistor lineal en el límite de la frecuencia infinita y como un resistor no lineal en el límite de la frecuencia cero. Independientemente de los mecanismos físicos que definen el estado del sistema, a muy bajas frecuencias el sistema tiene suficiente tiempo para ajustar su valor de resistencia a un valor momentáneo del parámetro de control

(corriente o tensión), de tal manera que el dispositivo se comporta como un resistor no lineal. Por otro lado, a muy altas frecuencias no hay suficiente tiempo para un cambio de resistencia durante un periodo de oscilaciones del parámetro de control, de tal manera que el dispositivo se comporta como un resistor lineal.

Se han encontrado bastantes sistemas que satisfacen las propiedades anteriores y se seguirán descubriendo muchos más que encajen en esta clasificación. En particular, el *comportamiento memresistivo* es una propiedad de los termistores, sistemas moleculares y nanoestructuras de película delgada.

Como ejemplo podemos destacar el modelo de un *sistema memresistivo* controlado por tensión usado por M. Di Ventra y Yu. V. Pershin. En este modelo el rango de variación de la resistencia del sistema (*R* varía entre los valores R_1 y R_2) está caracterizado por la constante α cuando $|V_M| \leq V_T$ y por la constante β cuando $|V_M| > V_T$, donde V_M es la tensión aplicada al *memresistor* y V_T es un umbral de tensión. Matemáticamente la respuesta de este *sistema memresistivo* se describe en las ecuaciones (37) y (38) con

$$G = x^{-1} \tag{39}$$

$$\dot{x} = -[\beta V_M + 0.5(\alpha - \beta) \cdot (|V_M + V_T| - |V_M - V_T|)] \cdot [\theta(V_M) \cdot \theta(x - R_1) + \theta(-V_M) \cdot \theta(R_2 - x)] \tag{40}$$

donde θ es la función de paso.

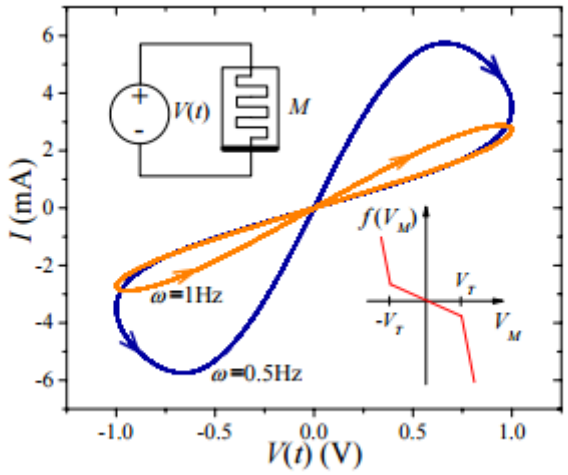

Fig. 17. Simulación de un *sistema memresistivo*. Esta gráfica se ha obtenido mediante una solución numérica de las ecuaciones (37), (38), (39) y (40) para el caso de un *sistema memresistivo M* conectado directamente a una fuente alterna de tensión $V(t) = V_0 sen(2\pi\omega t)$ usando los siguientes valores para los parámetros: R_1 = 20Ω, R_2 = 500Ω, V_T = 0.5V, α = -500V/(Ω·s), $\beta = 2\alpha$, V_0 = 1V.

Los resultados de la simulación expuestos en la Fig. 17 muestran un lazo de histéresis pellizcado y un colapso en la histéresis cuando se incrementa la frecuencia ω de la fuente alterna de tensión. [15]

El siguiente paso es extender las definiciones descritas a capacitancias e inductancias. Los dispositivos con memoria resultantes comparten muchas características de los *sistemas memresistivos*, pero tienen una diferencia fundamental: estos dispositivos almacenan energía. [14]

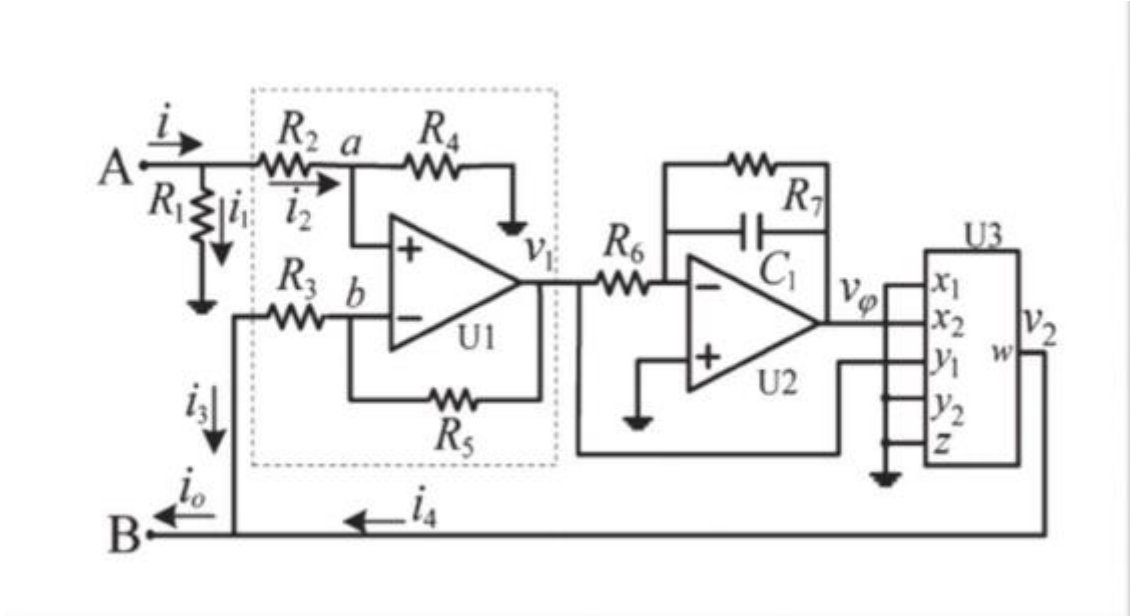

Fig. 18. Emulador de un *memresistor*.

En la Fig. 18 podemos observar el circuito emulador de un *memresistor* de tipo flotante, expresado en función y control del flujo magnético *φ*. El circuito posee siete resistores, un condensador, dos amplificadores operacionales (*U1* y *U2*) y un multiplicador analógico de señales (*U3*). [16]

6.2.3 – SISTEMAS MEMCAPACITIVOS

Definimos un *sistema memcapacitivo* controlado por tensión de orden *n* mediante las ecuaciones

$$q(t) = C(x, V_C, t) \cdot V_C(t) \tag{41}$$

$$\dot{x} = f(x, V_C, t) \tag{42}$$

donde *q(t)* es la carga del capacitor en el instante *t*, $V_C(t)$ es la tensión correspondiente, y *C* es la *memcapacitancia* que depende del estado del sistema. De igual modo podemos definir un *sistema memcapacitivo* controlado por carga de orden *n* mediante las ecuaciones

$$V_C(t) = C^{-1}(x, q, t) \cdot q(t) \quad (43)$$

$$\dot{x} = f(x, q, t) \quad (44)$$

donde C^{-1} es la inversa de la *memcapacitancia*.

Además, podemos definir una subclase de estos dispositivos que llamaremos *memcapacitores* controlados por tensión cuando las ecuaciones (41) y (42) se reducen a

$$q(t) = C \cdot \left[\int_{t_0}^{t} V_C(\tau) d\tau\right] \cdot V_C(t) \quad (45)$$

y un *memcapacitor* controlado por carga cuando las ecuaciones (43) y (44) se pueden expresar como

$$V_C(t) = C^{-1} \cdot \left[\int_{t_0}^{t} q(\tau) d\tau\right] \cdot q(t) \quad (46)$$

en las ecuaciones (45) y (46) el límite inferior de integración (tiempo inicial) se puede considerar -∞ si $\int_{-\infty}^{0} V_C(\tau) d\tau = 0$ o se puede considerar 0 si $\int_{-\infty}^{0} q(\tau) d\tau = 0$.

De la ecuación (41) podemos deducir que la carga es cero siempre que la tensión sea cero. De cualquier modo, debemos tener en cuenta que, en este caso, *I = 0* no implica *q = 0*, y por lo tanto este dispositivo puede almacenar energía.

Desde un punto de vista microscópico los cambios en la capacitancia pueden ocurrir por dos motivos: 1) debido a una variación en la forma estructural del sistema, o 2) por las propiedades mecánicas cuánticas de los portadores y cargas que componen el capacitor, o por ambos motivos.

Los efectos disipativos pueden estar involucrados en los cambios de la capacitancia del sistema tras la aplicación del parámetro externo de control. Estos procesos disipativos liberan energía en forma de calentamiento de los

materiales que componen el capacitor. Sin embargo, este calor no se puede considerar simplemente como una resistencia en serie con el capacitor.

De manera similar, puede haber situaciones en las que la energía no procede del parámetro de control, sino de las fuentes que controlan la ecuación de movimiento para la variable de estado, la ecuación (42) es necesaria para variar la capacitancia. Esta energía puede ser liberada en el circuito, amplificando así la corriente. Por lo tanto, las ecuaciones (41) y (42) para los *sistemas memcapacitivos* podrían describir dispositivos activos y pasivos.

Sin embargo, partiendo de un estado completamente descargado, la cantidad de energía retirada de un *sistema memcapacitivo* no puede exceder la cantidad de energía previamente añadida. Matemáticamente esta propiedad puede escribirse de la forma

$$U_C = \int_{t_0}^{t} V_C(\tau) \cdot I(\tau) d\tau \geq 0 \tag{47}$$

bajo la condición de que en $t = t_0$ no se almacene energía en el sistema. La ecuación (47) debe ser válida para cualquier forma del parámetro de control, tal como la tensión $V_C(t)$ aplicada al *sistema memcapacitivo* pasivo. Además, en los *sistemas memcapacitivos* con procesos disipativos (como el calentamiento), el signo de la ecuación (47) para $t > t_0$ no se logra (suponiendo $V_C(t) \neq 0$).

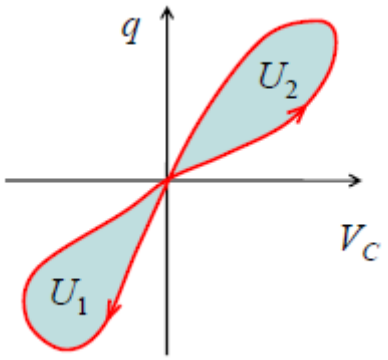

Fig. 19. Imagen de un lazo de histéresis pellizcado de un *sistema memcapacitivo*. La energía añadida o eliminada del sistema es el área entre la curva y el eje q. Las áreas de las regiones sombreadas U_1 y U_2 corresponden con la cantidad de energía añadida o eliminada en cada medio periodo. Los signos de U_1 y U_2 están determinados por la dirección del lazo. Para la dirección mostrada en la imagen, U_2 es positivo y U_1 es negativo.

La Fig. 19 muestra un lazo de histéresis de un *sistema memcapacitivo* que pasa por el origen. Las áreas sombreadas representan la energía U_i añadida o eliminada del sistema. Esta energía está asociada con cierto grado interno de libertad, es decir, relacionada con algunos procesos físicos elásticos o inelásticos que acompañan un cambio en la conductancia. El sistema es pasivo si $U_1 + U_2 = 0$, disipativo si $U_1 + U_2 > 0$ y activo si $U_1 + U_2 < 0$.

Por último, ya que la ecuación de estado (42) tiene una única solución en cualquier tiempo dado $t \geq t_0$, entonces si $V_C(t)$ es periódica con frecuencia ω, $V_C(t) = V_0 cos(2\pi\omega t)$, entonces la curva $q - V_C$ es un lazo simple que pasa por el origen, es decir, puede haber como máximo dos valores de la carga q para una tensión dada V_C si consideramos un dispositivo controlado por tensión, o dos valores de la tensión V_C para una carga dada q para un sistema controlado por carga. Este lazo es asimétrico con respecto al origen si, para el caso de las ecuaciones (41) y (42), $C(x, V_C, t) = C(x, -V_C, t)$ y $f(x, V_C, t) = f(x, -V_C, t)$.

Al igual que los *sistemas memresistivos*, un *sistema memcapacitivo* se comporta como un capacitor lineal en el límite de la frecuencia infinita, y como un capacitor no lineal en el límite de la frecuencia cero, suponiendo que las ecuaciones (42) y (44) admiten una solución en estado estacionario. El origen de este comportamiento se basa nuevamente en la capacidad de adaptarse a un cambio

lento de valor y a su incapacidad de responder a oscilaciones de frecuencia extremadamente altas.

Hay casos en los que se ha encontrado que la capacitancia *C* tiene una dependencia del tipo de histéresis en la tensión aplicada. Estos casos están relacionados con capacitores en nanoescala en los que **las trampas de interfaz o los nanocristales incrustados son los responsables de los efectos de memoria.**

A continuación, se presenta un ejemplo de un *sistema memcapacitivo* que se puede usar para simular supuestos experimentales en *memcapacitores*. Vamos a considerar un *sistema memcapacitivo* controlado por tensión de acuerdo con las ecuaciones (41) y (42). Este *sistema memcapacitivo C* está conectado a una fuente alterna de tensión $V(t) = V_0 sen(2\pi\omega t)$. La capacitancia *C* se elige para variar en el rango entre sus valores máximo y mínimo (C_1 y C_2) y cambia de acuerdo a las ecuaciones

$$C = x \tag{48}$$

$$\dot{x} = -[\beta V_C + 0.5(\alpha - \beta) \cdot (|V_C + V_T| - |V_C - V_T|)] \cdot [\theta(V_C) \cdot \theta(x - C_1) + \theta(-V_C) \cdot \theta(C_2 - x)] \tag{49}$$

donde V_C es la caída de tensión en el sistema memcapacitivo, V_T es una tensión de umbral, α es una tasa de variación para $|V_C| < V_T$ y β es una tasa de variación para $|V_C| > V_T$. La interpretación de los coeficientes α y β es la misma que en la Fig. 17. La respuesta del circuito se describe mediante la ecuación $V(t) = V_C = q/C$ donde *C* se puede hallar usando las ecuaciones (48) y (49).

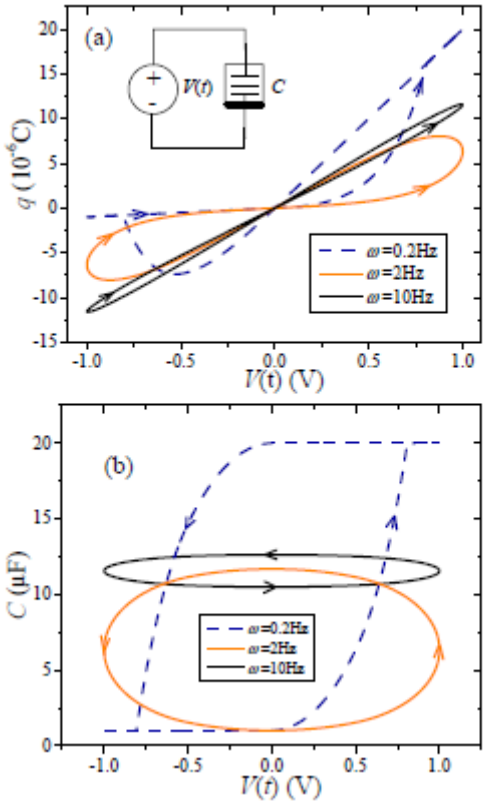

Fig. 20. Simulación de un circuito compuesto por un *sistema memcapacitivo* controlado por tensión y una resistencia. Se han utilizado los siguientes parámetros para los cálculos: C_1 = 1μF, C_2 = 20μF, V_T = 0.5V, α = -50μF/(V·s), β = 2α, V_0 = 1V.

La Fig. 20 representa los resultados de las simulaciones numéricas. Se pueden distinguir las características discutidas anteriormente como la histéresis de la capacitancia, el comportamiento a baja frecuencia y el lazo de histéresis pellizcado. También comprobamos la conservación de energía para el circuito dado y encontramos que para este modelo el *sistema memcapacitivo* se comporta como un dispositivo pasivo. [14] [15]

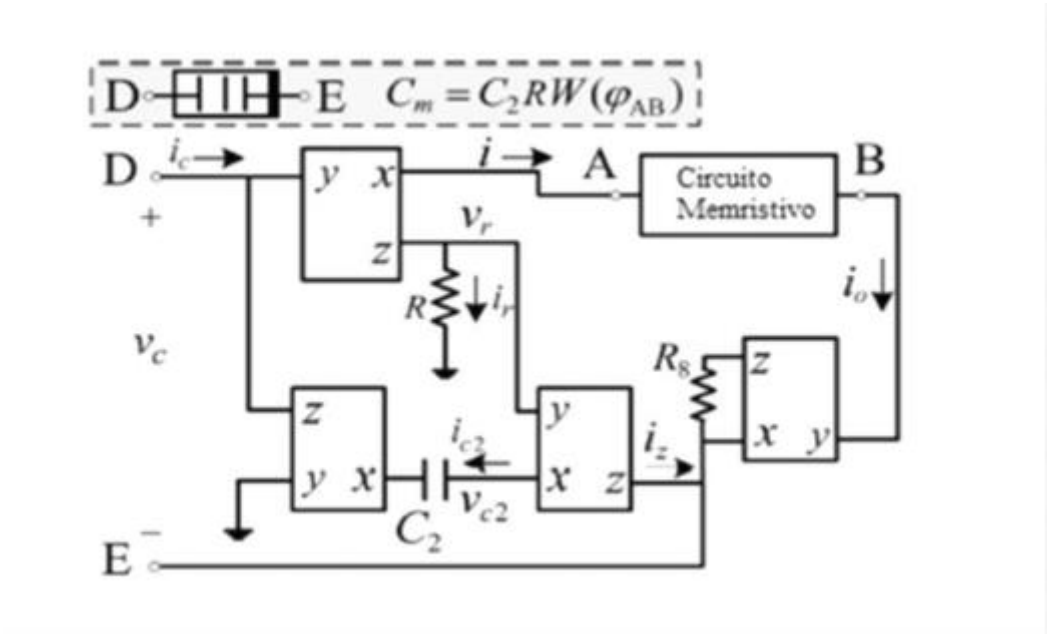

Fig. 21. Emulador de un *memcapacitor*.

En la Fig. 21 podemos observar el circuito emulador de un *memcapacitor* de tipo flotante. El circuito posee dos resistores, un condensador, cuatro convectores de corriente y el *circuito memresistivo* de la Fig. 18. [16]

6.2.4 – SISTEMAS MEMINDUCTIVOS

Para introducir esta tercera clase de dispositivos con memoria primero debemos definir el flujo

$$\phi(t) = \int_{-\infty}^{t} V_L(t')dt' \tag{50}$$

donde $V_L(t)$ es la tensión inducida en el inductor.

Definimos un *sistema meminductivo* controlado por corriente de orden *n* mediante las ecuaciones

$$\phi(t) = L(x, I, t) \cdot I(t) \tag{51}$$

$$\dot{x} = f(x, I, t) \tag{52}$$

donde *L* es la *meminductancia*, y definimos un *sistema meminductivo* controlado por flujo de orden *n* mediante

$$I(t) = L^{-1}(x, \phi, t) \cdot \phi(t) \tag{53}$$

$$\dot{x} = f(x, \phi, t) \tag{54}$$

siendo L^{-1} la inversa de la *meminductancia*.

También podemos definir una subclase de estos dispositivos que llamaremos *meminductores* controlados por corriente, donde las ecuaciones (51) y (52) se reducen a

$$\phi(t) = L \cdot \left[\int_{t_0}^{t} I(\tau) d\tau\right] \cdot I(t) \tag{55}$$

y un *meminductor* controlado por flujo cuando las ecuaciones (53) y (54) se pueden expresar como

$$I(t) = L^{-1} \cdot \left[\int_{t_0}^{t} \phi(\tau) d\tau\right] \cdot \phi(t) \tag{56}$$

En las ecuaciones (55) y (56) el límite inferior de integración se puede considerar -∞ si $\int_{-\infty}^{0} I(\tau) d\tau = 0$ o se puede considerar 0 si $\int_{-\infty}^{0} \phi(\tau) d\tau = 0$.

Extendiendo la definición de los *sistemas meminductivos* controlados por corriente, tomamos la derivada con respecto al tiempo en ambos lados de la ecuación (51)

$$V_L = \frac{d\phi}{dt} = L\frac{dI}{dt} + I\frac{dL}{dt} \tag{57}$$

El segundo término de esta ecuación (57) es una contribución adicional a la tensión inducida debido una *L* dependiente del tiempo. La energía almacenada en un *sistema meminductivo* controlado por corriente se puede calcular como

$$U_L(t) = \int_{t_0}^{t} V_L(\tau) \cdot I(\tau) d\tau = \int_{t_0}^{t} \left(L \frac{dI}{dt} + I \frac{dL}{dt} \right) \cdot I(\tau) d\tau \quad (58)$$

Cuando *L* es constante, obtenemos fácilmente la expresión para la energía $U_L = L \cdot I^2/2$. Normalmente esta U_L se interpreta como la energía del campo magnético generado por la corriente. De manera similar al criterio de pasividad de los *sistemas memcapacitivos* (ecuación (47)), la ecuación (58) proporciona el criterio de pasividad de un *sistema meminductivo* si en $t = t_0$ el sistema está en su estado energético mínimo y $U_L(t) \geq 0$ en cualquier momento.

Por último, como en el caso de los *sistemas memresistivos* y *memcapacitivos*, se espera que cuando el parámetro de control externo sea una función periódica en el tiempo con frecuencia ω, $I(t) = I_0 cos(2\pi\omega t)$, la curva $\varphi - I$ es un lazo simple que pasa por el origen, es decir, puede haber como máximo dos valores del flujo φ para una corriente *I* dada. Este lazo es asimétrico con respecto al origen si, para el caso de las ecuaciones (51) y (52), $L(x, I, t) = L(x, -I, t)$ y $f(x, I, t) = f(x, -I, t)$.

Además, la permeabilidad magnética puede ajustarse a las variaciones periódicas lentas de la corriente, mientras que no puede hacerlo a altas frecuencias, por lo que este dispositivo se comporta como un inductor no lineal a bajas frecuencias y como un inductor lineal a altas frecuencias.

En la electrónica, los inductores son principalmente un tipo de solenoide que consiste en una bobina de material conductor envuelta alrededor de un núcleo. La inductancia de un solenoide es proporcional a la permeabilidad relativa μ_r del material dentro del solenoide y también depende de los parámetros geométricos del sistema. **La forma** más sencilla **de introducir efectos de memoria** en un sistema de este tipo **es utilizar un material del núcleo cuya respuesta al campo magnético aplicado dependa de su historia**. Como ejemplo, podemos pensar en materiales ferromagnéticos que muestran una histéresis magnética como el hierro. Otra forma de introducir efectos de memoria es variando la forma del inductor.

Un modelo de circuito para un *sistema meminductivo* se puede formular de manera similar a los modelos de *sistemas memresistivos* y *memcapacitivos* descritos anteriormente. En particular, podemos imaginar una estructura cuya inductancia aumenta hasta un cierto valor cuando la corriente fluye en una dirección y disminuye hasta otro valor si se invierte la dirección del flujo de corriente. [14]

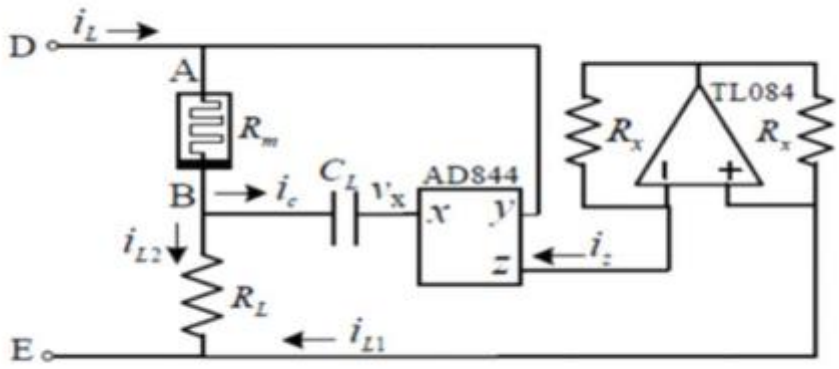

Fig. 22. Emulador de un *meminductor*.

En la Fig. 22 podemos observar el circuito emulador de un *meminductor* de tipo flotante. El circuito posee tres resistores, un condensador, un convector de corriente, un amplificador operacional y el *circuito memresistivo* de la Fig. 18. [16]

6.3 – CRITERIOS DE LA MEMCOMPUTACIÓN

Se han formulado unos criterios básicos que son necesarios y/o deseables para la implementación de la *memcomputación*. Algunos de ellos son similares a los introducidos por DiVincenzo en el campo de la *computación cuántica*, que a su vez son similares a los requeridos por cualquier paradigma de computación. Otros son específicos para la *memcomputación*. Como ya se ha mencionado, tanto la *memcomputación* como la *computación cuántica* se basan en el proceso en paralelo de información masiva. Sin embargo, los mecanismos básicos del proceso en paralelo de información masiva son muy diferentes. Mientras la

computación cuántica se basa en la superposición de estados, la *memcomputación* utiliza la dinámica colectiva de un gran número de sistemas (esencialmente clásicos). Sus criterios específicos son los siguientes.

- **Arquitectura escalable con proceso en paralelo de información masiva combinando procesamiento y almacenamiento de la información.** Fundamentalmente, la *memcomputación* es llevada a cabo por un circuito electrónico que contiene un conjunto de *memelementos* (*memresistores*, *memcapacitores*, *meminductores* o una combinación de todos ellos) que permiten el procesamiento y el almacenamiento de la información simultáneamente. Por definición, los *memelementos* almacenan la información en formato analógico en sus características de respuesta (además de la capacidad de los sistemas *memcapacitivos* y *meminductivos* para almacenar información en las energías de los campos eléctrico y magnético respectivamente). Se espera que todos, o al menos un gran número de *memelementos* estén involucrados en la computación en paralelo. Esta es la base de la ventaja potencial de la *memcomputación* sobre la computación tradicional secuencial.

  El procesamiento y almacenamiento de la información es una característica útil de los *memelementos*. Simplifica el diseño del hardware reduciendo la cantidad de componentes necesarios para realizar funciones de computación. De esta manera, es posible lograr densidades de integración más altas, así como conectividades más complejas. Claramente, ambos factores mejoran la funcionalidad del hardware. Por otra parte, se ha demostrado que el rendimiento de los circuitos lógicos mejora si varios tipos de *memelementos* se combinan juntos. Este hecho también debe tenerse en cuenta en el diseño de la arquitectura de la *memcomputación*.

- **Tiempos de almacenamiento suficientemente largos.** A continuación, consideraremos los requisitos que deben imponerse en los *memelementos*. En primer lugar, deben proporcionar tiempos de almacenamiento de información suficientemente largos. Por lo menos, mucho más largos que el tiempo de cálculo. Idealmente, se utilizarán *memelementos* con capacidades

de almacenamiento de memoria no volátil, tales como células emergentes de memoria no volátil. Es importante que muchos de estos elementos se realicen a nanoescala y, por lo tanto, muchos de estos elementos se pueden integrar en un solo chip.

Además, es deseable utilizar *memelementos* con bajo consumo de energía y tiempos cortos de lectura/escritura. Las células emergentes de memoria no volátil satisfacen estos requisitos y por lo tantos son candidatos ideales para arquitecturas de *memcomputación*.

- **Capacidad de inicializar estados de memoria**. Los *memelementos* deben ser inicializados antes de que comience el cálculo. La inicialización podría proporcionarse automáticamente en el caso de *memelementos* volátiles y puede no ser necesaria para *memelementos* de almacenamiento de valores intermedios o finales de los parámetros. En cualquier caso, el dispositivo de computación debe proporcionar un mecanismo para la inicialización de los *memelementos* relevantes. Se espera que esto sea una tarea fácil porque, típicamente, las funciones de respuesta de los dispositivos con memoria cambian entre dos valores limitantes. Por ejemplo, en el caso de los dispositivos *memresistivos* bipolares más estudiados, la aplicación de un impulso de alta amplitud de una polaridad dada durante un intervalo de tiempo suficientemente largo garantiza que el dispositivo cambie a uno de sus estados limitadores (dependiendo de la polaridad del impulso).

- **Mecanismo de dinámica colectiva, fuerte contenido de memoria.** La arquitectura del dispositivo debe proporcionar un mecanismo de dinámica colectiva en el que la evolución de un estado de un dispositivo con memoria depende de los estados de muchos o todos los demás dispositivos. Por ejemplo, en el caso de la *lógica memresistiva*, las tensiones aplicadas a un par de *dispositivos memresistivos* cambian el estado de uno de los dispositivos dependiendo del estado del segundo. Con el fin de obtener una conmutación fiable, también se requiere un fuerte contenido de memoria: las características del dispositivo en sus estados limitantes deben ser

suficientemente diferentes para proporcionar una influencia significativa en otros *memelementos*.

- **Capacidad de leer el resultado final de los *memelementos* relevantes.** Una vez que se realiza la computación, la lectura de la información debe realizarse (preferiblemente) sin modificar los estados de los *memelementos*. Esto se puede lograr si elegimos una entrada *u(t)* tal que el estado del dispositivo permanezca constante o varíe poco. Generalmente, esto no suele ser un problema en un sistema basado en *memelementos* con umbral. Esto también se puede lograr con dispositivos sin umbral, sin embargo, habrá que adaptar la entrada de lectura *u(t)* de manera apropiada para minimizar el cambio de estado.

- **Robustez contra pequeñas imperfecciones y ruido.** La insensibilidad a pequeñas imperfecciones de los componentes del ordenador o pequeños daños de la arquitectura debería ser una característica esencial de la mayoría de las arquitecturas de la *memcomputación*. De hecho, siempre se introducen pequeñas imperfecciones durante la fabricación de cada elemento y, por lo tanto, son inevitables. La arquitectura del ordenador debería ser robusta con respecto a dichas imperfecciones. Los daños son desviaciones más serias de la estructura ideal de computación. En el cerebro humano, por ejemplo, aunque varias neuronas mueren cada día, la funcionalidad cerebral global no se ve afectada hasta muchos años después. De manera que es razonable exigir que el funcionamiento de las redes de *memelementos* no sea sensible a daños relativamente pequeños. [13]

6.4 – MAQUINAS UNIVERSALES DE LA MEMCOMPUTACION

6.4.1 – INTRODUCCIÓN

Desde que Alan Turing inventó su máquina ideal en 1936, los matemáticos han sido capaces de desarrollar este concepto en lo que ahora se conoce como teoría de la complejidad computacional, una poderosa herramienta empleada para cuánto tiempo va a necesitar un algoritmo para resolver un problema con unos datos de entrada dados. Esta máquina ideal ahora se conoce como la máquina de Turing universal (UTM) y representa las bases conceptuales de todos los ordenadores digitales modernos.

La realización práctica de una UTM se hace usando la arquitectura von Neumann, la cual puede ser vista como un dispositivo que requiere una unidad central de procesamiento (CPU) que está físicamente separada de la memoria. La CPU contiene una unidad de control que dirige el funcionamiento de la máquina y todas las funcionas aritméticas y puertas lógicas que la máquina necesita durante la ejecución (unidad aritmético-lógica). Esta forma de cálculo requiere una gran cantidad de datos que se transferirán entre la CPU y la memoria, y esto limita la máquina tanto en tiempo como en energía.

La computación en paralelo puede mitigar estas limitaciones, pero no es capaz de resolver estos problemas: varios procesadores manipulan partes de todos los datos, trabajando con una memoria físicamente *cerrada*. Sin embargo, todos los procesadores tienen que comunicarse entre sí para resolver todo el problema, requiriendo una gran cantidad de transferencia de información entre ellos y sus memorias. Superar este problema de latencia de información requeriría una forma diferente de manipular y almacenar los datos.

La primera alternativa a la arquitectura von Neumann fue la arquitectura Harvard, desarrollada por H. Aiken. Las arquitecturas Harvard originales tenían memorias separadas para instrucciones y datos. Sin embargo, este término se utiliza hoy en día para referirse a máquinas con una única memoria principal, pero con instrucción separada y cachés de datos. Una alternativa más reciente es la arquitectura en pipeline, es decir, etapas de procesamiento de datos conectadas

en serie, donde la salida de una etapa es la entrada de la siguiente. Esta arquitectura se utiliza comúnmente en las CPU modernas y es particularmente eficiente para las unidades de procesamiento gráfico. Aunque algunos de estos conceptos han encontrado usos en la computación práctica, ninguna de estas alternativas resuelve completamente las limitaciones de la arquitectura von Neumann o muestra ventajas sustanciales en comparación con las máquinas de Turing.

Recientemente se ha propuesto un nuevo paradigma computacional, inspirado en la forma de operar de nuestro cerebro, que no se basa en el concepto UTM y que pone toda la carga de computación directamente en la memoria. Este paradigma es conocido como la *memcomputación*.

Al igual que el cerebro, las *máquinas de la memcomputación* realizarán cálculos con y en la memoria sin la necesidad de una CPU independiente. La memoria permite el aprendizaje y la capacidad de adaptación, eludiendo las conexiones rotas y la auto-organización de la computación en el camino de la solución, al igual que el cerebro es capaz de tener una cierta cantidad de daños y seguir funcionando a la perfección.

El concepto de la *memcomputación* puede realizarse en la práctica mediante la utilización de propiedades físicas de muchos materiales y sistemas que muestran cierto grado de memoria en sus funciones de respuesta a frecuencias y fortalezas particulares de las entradas. De hecho, se ha visto una creciente oleada de actividades con *memelementos* y su realización real en una variedad de sistemas. Por ejemplo, se han estudiado propiedades físicas y estadísticas, capacidad de cálculo, y más aspectos para redes de *memresistores*. Sin embargo, pueden surgir dinámicas más complejas y propiedades muy interesantes utilizando *memelementos* distintos de los *memresistores* y combinándolos con dispositivos electrónicos tradicionales. Por ejemplo, se ha realizado una implementación real de una *máquina de memcomputación* que tiene la misma arquitectura de una memoria dinámica de acceso aleatorio (DRAM), pero emplea *memcapacitores* para calcular y almacenar información. Esta arquitectura en particular es solo un ejemplo de lo que se puede lograr con los *memelementos*. Sin embargo, ya muestra dos características que no están

disponibles en nuestros ordenadores modernos: el *paralelismo intrínseco* y el *polimorfismo funcional*.

La primera característica significa que todos los procesadores de memelementos operan simultáneamente y colectivamente durante el cálculo. De esta manera, los problemas que de otro modo requerirían varios pasos para ser resueltos se pueden ejecutar en uno o unos pocos pasos. La segunda característica se refiere a la capacidad de calcular diferentes funciones sin modificar la topología de la red de la máquina, simplemente aplicando las señales de entrada apropiadas. Este polimorfismo, que es similar al de nuestro cerebro y algunos de nuestros órganos, también muestra otra característica importante de las *máquinas de la memcomputación*: su unidad de control, que no es una CPU completa, ya que no necesita ninguna unidad aritmético/lógica, y que es necesaria para controlar la ejecución del tipo de problema que necesita ser resuelto por la máquina, puede ser alimentada directamente por los datos de entrada, o indirectamente a través del flujo de datos de entrada a los *memprocesadores*.

La última propiedad importante de las UMM, denominada *superposición de la información*, está relacionada con la manera en que los *memprocesadores* interconectados físicamente pueden almacenar una cantidad de datos. Más específicamente, la *superposición de la información* es la capacidad de una red interconectada de *memprocesadores* de almacenar y comprimir una cantidad de información mayor de la que es posible con *memprocesadores* no interconectados. [17]

6.4.2 – MEMPROCESADORES

La unidad constitutiva básica de una *arquitectura de la memcomputación* es lo que llamamos *memprocesador*. Aquí, se presenta una definición formal de *memprocesador* que en realidad representa el vínculo entre las *arquitecturas reales de la memcomputación* y la definición formal de una UMM. Esto también debería aclarar que **un *memprocesador* no está necesariamente hecho de *memelementos*, sino que es un objeto mucho más general.**

Definimos un *memprocesador* como un objeto definido por cuatro variables *(x, y, z, σ)* donde *x* es el estado del *memprocesador*, *y* es la matriz de variables internas, *z* la matriz de variables que conectan desde un *memprocesador* a otros *memprocesadores* y *σ* es un operador que define la evolución.

$$\sigma[x, y, z] = (x', y') \tag{59}$$

Cuando dos o más *memprocesadores* están conectados, tenemos una red de *memprocesadores* (memoria computacional). En este caso definimos el vector *x* como el estado de la red (es decir, la matriz de todos los estados x_i de cada memprocesador), y $z = U_i\, z_i$ será la matriz de todas las variables de conexión, con z_i como la matriz de conexión de variables del *memprocesador i*-ésimo.

Sean z_i y z_j los vectores de las variables de conexión de los memprocesadores *i* y *j*, entonces si $z_i \cap z_j \neq 0$ decimos que los dos *memprocesadores* están conectados. Alternativamente, un *memprocesador* no está conectado a ningún otro *memprocesador* cuando *z* está completamente determinado por *x* e *y*, y

$$\sigma[x, y, z(x, y)] = (x, y) \tag{60}$$

lo que significa que el *memprocesador* no tiene variaciones.

Una red de *memprocesadores* tiene la evolución de las variables de conexión *z* dado por el operador de evolución *Ξ* definido como

$$\Xi[x, y, z, \mathfrak{s}] = z' \tag{61}$$

Donde $y = U_i\, y_i$ y $\mathfrak{s}$ es la matriz de las señales externas que se pueden aplicar a un subconjunto de conexiones para proporcionar estímulos a la red. Finalmente, la evolución completa de la red se define por el sistema

$$\begin{cases} \sigma[x_1, y_1, z_1] = (x_1', y_1') \\ \vdots \\ \sigma[x_n, y_n, z_n] = (x_1', y_1') \\ \Xi[x, y, z, \mathfrak{s}] = z' \end{cases} \tag{62}$$

Los operadores de evolución σ y Ξ pueden interpretarse como operadores de evolución discreta o continua. La interpretación del operador de evolución discreta incluye también las redes neuronales artificiales, mientras que la interpretación continua del operador representa sistemas dinámicos más generales. Se analizan dos tipos de operadores continuos: los operadores que representan *memprocesadores* compuestos sólo por *memelementos*, y *memprocesadores* compuestos por dispositivos electrónicos genéricos. [17]

6.4.2.1 – MEMPROCESADORES COMPUESTOS POR MEMELEMENTOS

Las ecuaciones estándar de un *memelemento* vienen dadas por las siguientes relaciones

$$x_{ex}(t) = g(x_{in}, u, t)u(t) \tag{63}$$

$$\dot{x}_{in}(t) = f(x_{in}, u, t) \tag{64}$$

donde x_{in} denota un conjunto de variables de estado que describen el estado interno del sistema, u y x_{ex} son dos variables constitutivas complementarias que denotan la entrada y salida del sistema, g es una respuesta generalizada, y f es una función vectorial continua. En este caso las variables de conexión z son $x_{ex}(t)$ y $u(t)$ y el estado x coincide con $x_{in}(t)$. Ahora, la ecuación (63) se puede interpretar como una restricción para la ecuación (64) en el sentido de que podemos expresar $u = u(x_{in}(t), x_{ex}(t))$ como solución de $x_{ex}(t) = g(x_{in}, u, t)u(t)$, de modo que podemos reemplazar $u(x_{in}, y)$ en la ecuación (64) y olvidarnos de la ecuación (63). En este caso la evolución en un tiempo T de la ecuación (64) es dada por

$$x_{in}(t+T) - x_{in}(t) = \int_t^{t+T} f\big(x_{in}(\tau), u\big(x_{in}(\tau), x_{ex}(\tau)\big), \tau\big)d\tau \tag{65}$$

y definiendo $x = x_{in}(t)$ y $x' = x_{in}(t+T)$ tenemos

$$\sigma[x, y, z] = \sigma[x, y] = x_{in}(t) + \int_t^{t+T} f\big(x_{in}(\tau), u\big(x_{in}(\tau), x_{ex}(\tau)\big), \tau\big)d\tau \tag{66}$$

Por otro lado, el operador Ξ está definido por las leyes de Kirchhoff y los generadores externos para incluir las señales externas s. [17]

6.4.2.2 – MEMPROCESADORES COMPUESTOS POR DISPOSITIVOS ELECTRÓNICOS GENÉRICOS

En este caso, la ecuación que define un memprocesador es una ecuación algebraica diferencia que representa cualquier circuito electrónico. Puede ser modelado como

$$\frac{d}{dt}q(x,y,z) = f(x,y,z,t) \tag{67}$$

donde *(x, y, z)* representan todas las variables de estado del circuito.

Usando la regla de la cadena tenemos $\frac{dq}{dt} = \frac{\partial q}{\partial x}\dot{x} + \frac{\partial q}{\partial y}\dot{y} + \frac{\partial q}{\partial z}\dot{z}$. Mediante la teoría de circuitos y el análisis de nodos siempre existe una elección de y para que la matriz jacobiana

$$J_{x,y}(x,y,z) = \begin{bmatrix} \frac{\partial q(x,y,z)}{\partial x} & \frac{\partial q(x,y,z)}{\partial y} \end{bmatrix} \tag{68}$$

sea cuadrada pero no necesariamente invertible. Si $J_{x,y}$ no es invertible, podemos eliminar algunas variables incluyendo restricciones como en la sección anterior obteniendo así un $J_{x,y}$ reducido que es invertible. Por lo tanto, podemos suponer $J_{x,y}$ invertible y mediante la regla de la cadena y las ecuaciones (67) y (68)

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J_{x,y}^{-1}(x,y,z)\left(f(x,y,z,t) - \frac{\partial q(x,y,z)}{\partial z}\dot{z}\right) \tag{69}$$

La evolución en un tiempo T viene dada por

$$\sigma[x,y,z] = \begin{bmatrix} x(t+T) \\ y(t+T) \end{bmatrix} = \tag{70}$$

$$= \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \int_{t}^{t+T} J_{x,y}^{-1}(x,y,z)\left(f(x,y,z,t) - \frac{\partial q(x,y,z)}{\partial z}\dot{z}\right)d\tau$$

También en este caso el operador Ξ estará definido por las leyes de Kirchhoff y los generadores externos para incluir las señales externas s. [17]

6.4.3 – MÁQUINA UNIVERSAL DE LA MEMCOMPUTACIÓN

Ahora podemos introducir el concepto general de una UMM. **La UMM es una máquina ideal formada por un conjunto de memprocesadores capaces de realizar operaciones digitales** (lógicas) **o analógicas** (funcionales) **controlados por una unidad de control.** La computación con y en memoria se puede describir de la siguiente forma. **Cuando dos o más memprocesadores están conectados, a través de una señal enviada por la unidad de control, los memprocesadores cambian sus estados internos de acuerdo con sus estados iniciales y la señal**, dando lugar al *paralelismo intrínseco* y al *polimorfismo funcional* de los que hemos hablado anteriormente. [17]

6.4.3.1 – DEFINICIÓN FORMAL

Definimos la UMM como

$$UMM = (M, \Delta, P, S, \Sigma, p_0, s_0, F) \tag{71}$$

donde *M* es el conjunto de estados posibles de un *memprocesador*. Puede ser un conjunto finito M_d o un conjunto infinito de estados discretos M_a, por lo tanto, *M* puede expresarse como $M = M_d \oplus M_a$. Δ es un conjunto de funciones

$$\delta_a : M^{m_\alpha} \backslash F \times P \rightarrow M^{m'_\alpha} \times P^2 \times S \tag{72}$$

donde $m_\alpha < \infty$ es el número de *memprocesadores* utilizados como entrada de la función δ, y $m'_\alpha < \infty$ es el número de *memprocesadores* utilizados como salida por la función δ_α; P es el conjunto de matrices de punteros p_α que selecciona el *memprocesador* deseado por δ_α y S es el conjunto de índices α; Σ es el conjunto de estados iniciales escritos por el dispositivo de entrada en la memoria computacional; $p_0 \in P$ es la matriz inicial de punteros; s_0 es el índice inicial α y $F \subseteq M$ es el conjunto de estados finales.

Para una mayor claridad indicamos que, ya que p_α, p'_α, $p_\beta \in P$ y las matrices $p_\alpha = \{i_1, \dots, i_{m_\alpha}\}$, $p'_\alpha = \{j_1, \dots, j_{m'_\alpha}\}$ y $p_\beta = \{k_1, \dots, k_{m_\beta}\}$ y $\beta \in S$, entonces la función δ_α puede expresarse como

$$\delta_\alpha[x(p_\alpha)] = \left(x'(p'_\alpha), \beta, p_\beta\right) \tag{73}$$

donde x es el vector de estados de los *memprocesadores*, así $x(p_\alpha) \in M^{m_\alpha}$ son los estados de los *memprocesadores* seleccionados como entrada por δ_α, mientras que $x'(p'_\alpha) \in M^{m'_\alpha}$ son los estados de salida de δ_α. Entonces δ_α lee los estados $x(p_\alpha)$ y escribe los nuevos estados $x'(p'_\alpha)$, y al mismo tiempo prepara el nuevo puntero p_β para la siguiente función δ_β con entrada $x'(p_\beta) \in M^{m_\beta}$.

Hay que tener en cuenta que las dos características importantes de las UMM están incluidas en la definición del conjunto de funciones δ_α. De hecho, las UMM, a diferencia de las UTM, pueden tener más de una función de transición δ_α (*polimorfismo funcional*) y cualquier función δ_α actúa simultáneamente en un conjunto de *memprocesadores* (*paralelismo intrínseco*). Otra diferencia de la UTM es que la UMM no distingue entre los estados de la máquina y los símbolos escritos en la cinta. Por el contrario, esta información está completamente codificada en los estados de los *memprocesadores*. Esto es un ingrediente crucial para construir una máquina capaz de procesar y almacenar información en la misma plataforma física.

Otra observación importante es que, a diferencia de una UTM que tiene un número finito de estados discretos y una cantidad ilimitada de almacenamiento en cinta, una UMM puede operar en un número infinito de estados continuos, aunque el número de *memprocesadores* sea finito. La razón es que cada

memprocesador es un dispositivo analógico con un conjunto de valores de estado continuos.

Por último, se puede observar que la definición formal del *memprocesador* de la ecuación (59) y la red de *memprocesadores* de la ecuación (62) es compatible con la función δ_α definida en las ecuaciones (72) y (73). De hecho, la topología y evolución de la red está asociada con el estímulo $\mathcal{s}$, mientras que la unidad de control define todas las posibles $\delta_\alpha \in \Delta$ en el sentido de que pueden obtenerse aplicando una cierta señal $\mathcal{s}_\alpha$ a la red. La evolución de la red determina x' mientras que β y p_β son definidas por la unidad de control para la siguiente etapa del procesamiento.

Esta descripción física señala una peculiaridad relevante de las UMM. Con el fin de implementar la función δ_α la unidad de control manda el estímulo $\mathcal{s}$ a $x(p_\alpha)$, y la red de *memprocesadores* bajo ese estímulo cambian los estados de $x(p'_\alpha)$ a $x'(p'_\alpha)$. Incluso si $dim(p_\alpha) = 1$, que es el caso en el que la unidad de control actúa en un único *memprocesador*, tenemos *paralelismo intrínseco*. [17]

6.4.3.2 – PRUEBA DE UNIVERSALIDAD

Para demostrar la universalidad de las UMM se proporciona una prueba implícita: probar que una UMM puede simular cualquier UTM, siendo esta una condición suficiente de universalidad.

Primero tomamos una UMM con los estados del *memprocesador* definidos por $M = Q \cup \Gamma$. Una célula de memoria se localiza con el puntero j_s mientras que el resto de células se localizan con el puntero $j = \ldots, -k, \ldots, -1, 0, 1, \ldots, k, \ldots$. Definimos la matriz de punteros p, definida por $p = \{j_s, j\}$. Utilizamos la célula j_s para codificar el estado $q \in Q$, y en las otras células codificamos los símbolos en Γ.

Tomamos Δ, compuesta por una única función $\bar{\delta}[x(p)] = (x'(p), p')$, donde hemos suprimido el índice de salida β porque la función es única. Los nuevos estados x' son escritos por $\bar{\delta}$ de acuerdo con la tabla de las UTM, y en particular en $x'(j_s)$

encontramos el nuevo estado que tendrá la UTM, y en *x'(j)* encontramos el símbolo que la UTM escribirá en la cinta. Finalmente, el nuevo puntero *p'* es dado por $p' = \{j_s, j'\}$ donde $j' = j$ si δ de la UTM no requiere desplazamientos, $j' = j + 1$ si hay desplazamiento a la derecha y $j' = j - 1$ si hay desplazamiento a la izquierda. Finalmente, hay que escribir en $x(j_s)$ el estado inicial q_0 y los símbolos iniciales Σ donde sea necesario, la UMM con $\Delta = \bar{\delta}$ simula la UTM con δ.

Así se demuestra que una UMM es una Turing-completa que puede simular cualquier máquina de Turing. Hay que tener en cuenta que al contrario no es necesariamente cierta esta afirmación. [17]

6.5 – SOLUCIÓN DE PROBLEMAS NP-COMPLETOS

6.5.1 – DEMOSTRACIÓN TEÓRICA

Consideramos problemas NP-completos a aquellos cuyo algoritmo de resolución requiere un árbol de solución que crece exponencialmente con la dimensión *n* de la entrada. Este algoritmo, implementado en una máquina de Turing determinista requeriría un tiempo exponencial (pasos) con respecto a *n* para poder ser resuelto.

Consideramos una descripción formal del árbol de solución asumiendo que la rama del árbol que lleva a la solución necesita un número polinómico de iteraciones con respecto a *n*. En cada iteración *i* podemos asumir que cada rama se divide en M_i nuevas ramas, así que el árbol de solución completo tiene un número total de nodos $N_{nodos} \leq \sum_{k=1}^{N_s} \prod_{i=1}^{k} M_i$.

Siguiendo esta fórmula, si el árbol más simple tiene un número medio de $\bar{M}$ divisiones de ramas, entonces el tiempo requerido por una máquina de Turing para explorar el árbol de solución, que es proporcional al número de nodos, es del orden de $\bar{M}^{N_S} = \bar{M}^{P(n)}$. Por el contrario, se puede probar que la UMM puede encontrar la solución en un tiempo proporcional a *P(n)*.

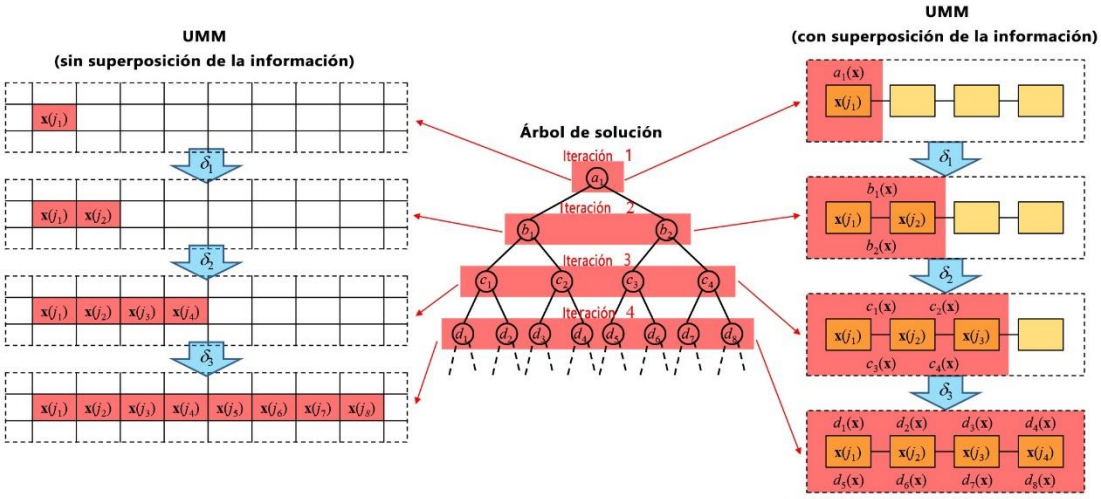

Fig. 23. Árbol de solución de un problema NP-completo implementado en una UMM. La transición de la iteración i a la iteración i + 1 del árbol de solución es calculada por la función $\delta_\alpha = \delta_i$, actuando en el grupo de memprocesadores que codifican la iteración i para hallar la iteración i + 1 al mismo tiempo. Las UMM que no emplean la superposición de la información codifican la información de cada nodo del árbol de solución en un número proporcional de memprocesadores, mientras que las UMM que emplean la superposición de la información pueden codificar la información de un grupo de nodos en un estado colectivo de un grupo de memprocesadores.

Para ello consideramos que la UMM opera de la siguiente manera: La unidad de control envía una señal de entrada a un grupo de *memprocesadores* interconectados que codifican la iteración *i* del árbol de solución para calcular la iteración *i+1* a la vez. De este modo, todo el proceso de cálculo se puede formalizar siguiendo el esquema de la Fig. 23. Consideramos un conjunto ***Δ*** de funciones δ_α con $\alpha = 1, \ldots, P(n)$ tal que $\delta_\alpha[x(p_\alpha)] = (x'(p_{\alpha+1}), \alpha + 1, p_{\alpha+1})$ donde $x(p_\alpha)$ codifica todos los nodos correspondientes a la iteración $\alpha = i$ del árbol de solución en, a lo sumo, $dim(p_\alpha) \leq \prod_{i=1}^{\alpha} M_i$ *memprocesadores*, y $x'(p_{\alpha+1})$ codifica todos los nodos correspondientes a la iteración $\alpha + 1 = i + 1$ del árbol de solución en, a lo sumo, $dim(p_{\alpha+1}) \leq \prod_{i=1}^{\alpha+1} M_i$ *memprocesadores*. Utilizando la característica de la *superposición de la información*, los datos codificados en $N_{nodos\beta} \leq \prod_{i=1}^{\beta} M_i$ pueden ser codificados en un número de *memprocesadores* $dim(p_\beta) \leq N_{nodos\beta}$.

Ya que δ_α corresponde a un único paso de cálculo para la UMM, se necesitará un tiempo proporcional a $P(n)$ para encontrar la solución. Por lo tanto, en principio, la UMM puede resolver un problema NP-completo en un tiempo

proporcional a *P(n)*, es decir, la UMM tiene la misma potencia computacional que una máquina de Turing no determinista (NTM).

Usando este esquema simplificado para la solución de un problema NP-completo, podemos destacar la diferencia fundamental entre la UMM y la NTM. La NTM necesita, en principio, un número de unidades de procesamiento y un número de cintas que crecen exponencialmente para poder explorar todos los caminos de solución posibles. Por el contrario, **una UMM**, para poder encontrar una solución de un problema NP-completo en un tiempo polinómico, **necesita un número de *memprocesadores* que, a lo sumo, crecen exponencialmente**, ya que se puede procesar una iteración completa del árbol de solución en un solo paso. Sin embargo, **la *superposición de la información* nos permite reducir el número de *memprocesadores* que crecen exponencialmente de una manera sustancial**, haciendo que la implementación práctica de una UMM sea más sencilla. Hay que tener en cuenta que esta prueba es válida para cualquier problema cuya solución se pueda transformar en un árbol que crece exponencialmente, ya sea un problema NP-completo o NP-hard. [17]

6.5.2 – DEMOSTRACIÓN PRÁCTICA

6.5.2.1 – INTRODUCCIÓN

Para la demostración práctica de la resolución de problemas NP-completos se procede a **resolver un laberinto con una red de *memresistores***. Esta red se complementará con otros dispositivos electrónicos clásicos para formar un *memprocesador*. El *memprocesador* requiere únicamente un paso para encontrar la solución al laberinto.

Merece la pena destacar que también se podría implementar la resolución del laberinto con redes de *memcapacitores* o *meminductores*. Se elige trabajar con *memresistores* por su mayor simplicidad. [18]

6.5.2.2 – MEMPROCESADOR

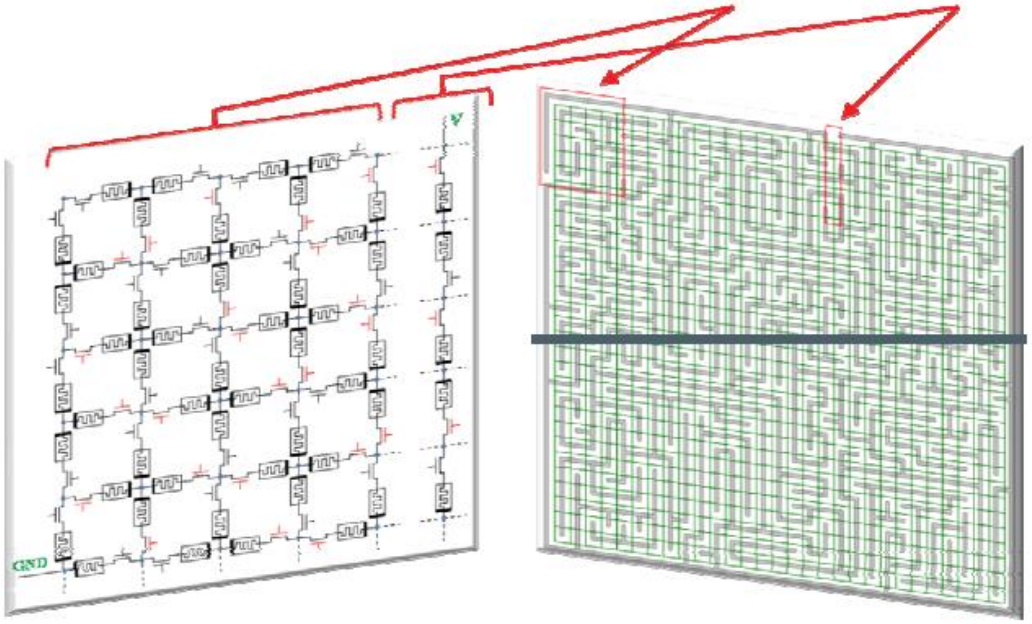

Fig. 24. Mapeado de un laberinto en una red de memresistores.

Empezamos mapeando un laberinto en una red de *memresistores*, como se muestra en la Fig. 24. Lo primero es superponer periódicamente matrices de líneas horizontales y verticales en el laberinto. El periodo de estas matrices corresponde al periodo intrínseco del laberinto. Los puntos de cruce de las líneas horizontales y verticales definen los puntos de la rejilla de la red de *memresistores*. La red se compone de *memresistores* y switches que se conectan en los puntos de la rejilla. Ya que la dirección del flujo de corriente no es conocida, la polaridad de los *memresistores* adyacentes se elige para ser alternante. Se asume que las señales externas se pueden aplicar en cualquier punto de la rejilla con el propósito de inicializar los estados de los *memresistores,* así como para leer los resultados del cálculo. Los switches controlados externamente se usan para definir la topología del laberinto: una pared del laberinto se consigue con un switch en el estado “no conectado”. Esta arquitectura permite modelar distintos laberintos con el mimo *memprocesador* sin la necesidad de fabricar un *memprocesador* específico para cada laberinto.

La inicialización del *memprocesador* se puede hacer mediante la aplicación simultánea de *GND* y V_1, con un patrón de tablero de ajedrez, en todos los puntos de la rejilla de la red de *memresistores*. El cálculo realizado por este *memprocesador* ocurre cuando una tensión constante *V* se aplica a través de los dos puntos de la rejilla correspondientes a los puntos de entrada y salida del laberinto. En este caso, la corriente fluye solo por aquellos *memresistores* que conectan la entrada y la salida. El estado de estos *memresistores* cambia debido a la corriente, así el laberinto se resuelve de manera masiva en paralelo, ya que todos los *memresistores* de la red participan simultáneamente en los cálculos. Asumiendo que en el momento inicial todos los *memresistores* fueron inicializados en el estado de alta resistencia, cuando pasa el tiempo, todos los *memresistores* del camino de solución cambian su resistencia, cambiando al estado de baja resistencia. De hecho, la cadena de *memresistores* en el estado "ON" que conecta los puntos de entrada y salida representa la solución del laberinto.

Si existen múltiples soluciones, entonces la más corta contendrá menos *memresistores* y ofrecerá menor resistencia que la más larga. [18]

6.5.2.3 – MODELO NUMÉRICO

El modelo numérico de una red de *memresistores* es fácil de implementar y ofrece por sí mismo un algoritmo práctico de computación para la resolución de laberintos. Esta aproximación computacional, sin embargo, requiere múltiples pasos, al contrario del *memprocesador* real que sólo necesita un único paso para realizar la operación.

Para mayor claridad, se utiliza un modelo simple de un *memresistor* cuya *memresistencia* es dada por

$$R_{ij}^{M} = R_{ON x_{ij}} + R_{OFF}\left(1 - x_{ij}\right) \tag{74}$$

donde R_{ON} y R_{OFF} son los valores mínimo y máximo de la *memresistencia*, x_{ij} es la variable de estado interna adimensional vinculada a la región $0 \leq x_{ij} \leq 1$ y (i, j) son los índices de la rejilla del *memresistor* para identificar su localización en la red. La dinámica de x_{ij} viene dada por

$$\frac{dx_{ij}}{dt} = \alpha I_{ij}(t) \tag{75}$$

donde α es una constante y $I_{ij}(t)$ es la corriente que fluye por el memresistor (ij). En cada paso de tiempo, el potencial de todos los puntos de la rejilla se calcula con las leyes de Kirchhoff. El cambio en los estados de los *memresistores* se calcula usando la ecuación (75). [18] [19]

6.5.2.4 – RESULTADOS NUMÉRICOS

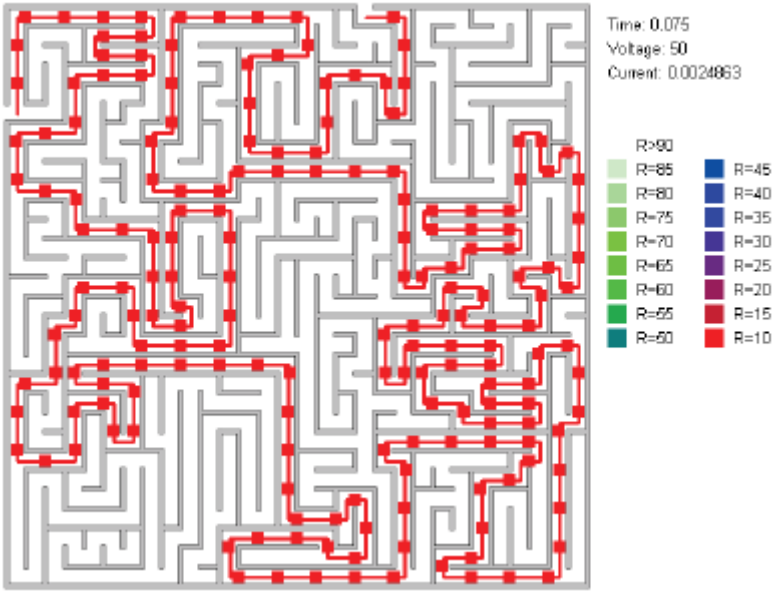

Fig. 25. Solución de un laberinto de un único camino. Estado de la red en $t = 0.075$ *s*. La cadena de *memresistores* en estado de baja resistencia conecta los puntos de entrada y salida del laberinto.

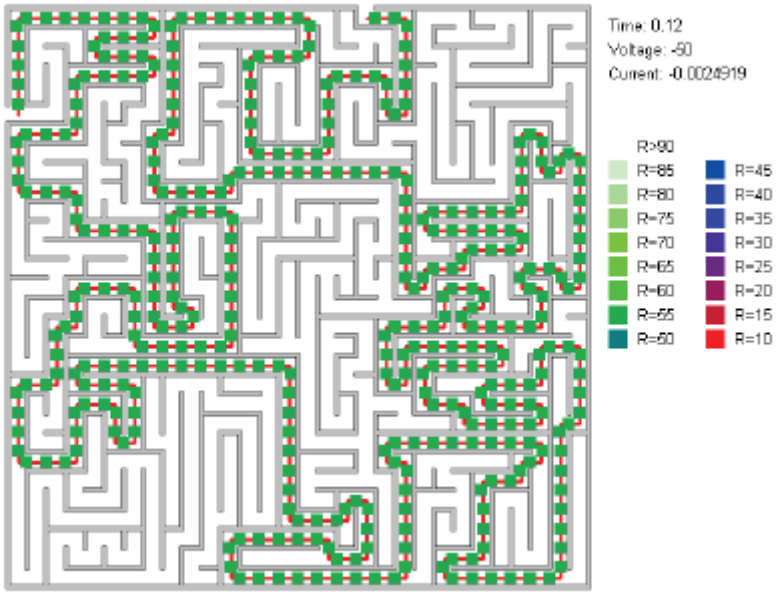

Fig. 26. Solución de un laberinto de un único camino. Estado de la red en t = 0.12 s. La cadena de *memresistore*s en estado de baja resistencia conecta los puntos de entrada y salida del laberinto.

Todos los resultados numéricos se han obtenido usando los parámetros R_{ON} = *10 Ω*, R_{OFF} = *100 Ω*, $R_{ij}^{M}(t=0)$ = *91 Ω*, *x(t=0) = 0.1* y $\alpha = 10^4/(s \cdot A)$ para todos los *memresistores*. La tensión aplicada es de *50 V* durante el primer intervalo de tiempo de *0.1 s*, y de *-50 V* para *t > 0.1 s*. El signo de la tensión aplicada varía para representar la solución del laberinto como en la Fig. 25 y la Fig. 26.

Los laberintos se han mapeado con un *memprocesador* de *n x n* puntos con *n = 30*. En este caso n^2 nos da el número total de puntos de la rejilla.

La Fig. 25 y la Fig. 26 presentan los resultados de la solución de un laberinto con un único camino. Con los parámetros seleccionados se requiere aproximadamente 0.06 s para cambiar los *memresistores* a lo largo del camino de la solución al estado de baja resistencia "ON". **La secuencia resultante de *memresistores* en estado de baja resistencia representa la solución del laberinto**, como se muestra en la Fig. 25. Esta solución se puede apreciar si cambiamos el signo de la tensión aplicada y esperamos. La resistencia de los *memresistores* que se encuentran en estado "ON" aumentará y la resistencia de los *memresistores* que se encuentran en estado "OFF" disminuirá, como podemos observar en la Fig. 26. En este momento, todos los *memresistores* que

se encuentran en el camino de la solución se encuentran en el mismo estado y la solución es más visible.

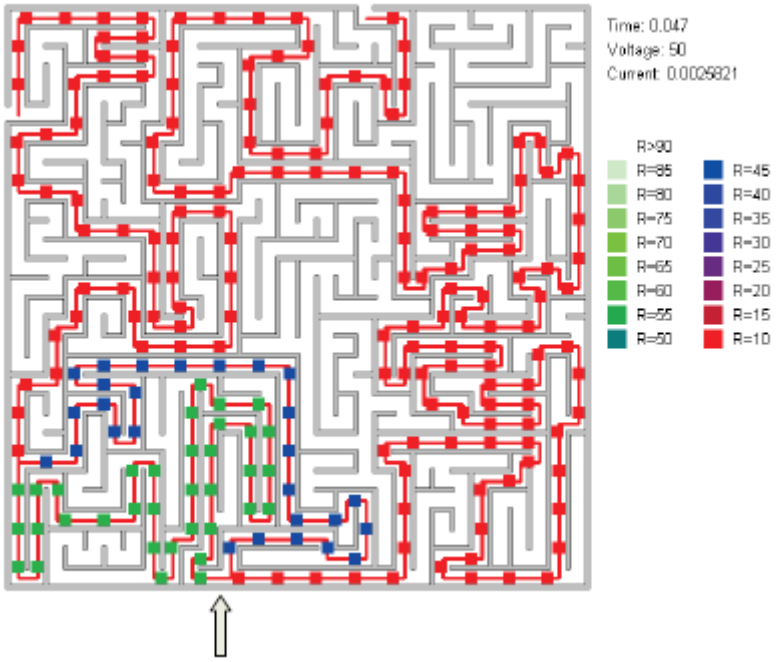

Fig. 27. Solución de un laberinto con múltiples caminos. Estado de la red en t = 0.047 s. La solución del laberinto contiene dos segmentos comunes y dos segmentos alternativos de diferentes longitudes en la esquina inferior izquierda. La memresistencia en el camino más corto (azul) es más pequeña que en el camino más largo (verde), ya que la corriente que atraviesa el camino más corto es mayor. La flecha indica dónde se ha modificado el laberinto con respecto a la Fig. 25 y la Fig. 26.

La Fig. 27 muestra un laberinto con dos caminos de solución. En este caso, la corriente fluye por un segmento común que se divide entre dos posibles caminos. Los *memresistores* que se encuentran en el segmento común cambian su resistencia más rápidamente que los que se encuentran en los dos posibles caminos. Comparando los *memresistores* de los dos posibles caminos, los *memresistores* del camino más corto cambian su resistencia más rápido que los del camino más largo. Este hecho nos permite filtrar todas las posibles soluciones de acuerdo con su longitud, de tal manera que **las resistencias de los *memresistores* de los caminos más cortos son más pequeñas**. [18]

7 – CONCLUSIONES

La *computación cuántica* es la alternativa a la computación clásica que está más cerca de poder ser una realidad asequible, viable y práctica. Parece una cuestión de tiempo que se pueda obtener un *ordenador cuántico* capaz de sustituir a nuestros ordenadores actuales.

Si observamos los *ordenadores cuánticos* fabricados a día de hoy, la apariencia de los mismos nos recuerda a los primeros ordenadores fabricados.

Fig. 28. Comparación de los primeros ordenadores con los ordenadores cuánticos actuales.

Por otro lado, existe el problema de la falta de *algoritmos cuánticos*. Esto quiere decir que hace falta diseñar todo el sistema desde el principio. Así que el camino de la *computación cuántica* es todavía largo.

En cuanto a la resolución de problemas computacionales, la *computación cuántica* es capaz de resolver esos problemas de manera más rápida al utilizar la superposición de estados, sin embargo, al tratarse de un modelo de computación en desarrollo, requiere de grandes recursos que hacen su viabilidad se vea reducida.

Debido a la lenta evolución de la *computación cuántica* y a la cercanía de alcanzar los límites atómicos de la computación clásica, no debemos descartar otras alternativas como la *memcomputación*.

Como hemos podido observar, la *memcomputación* es una idea que se encuentra en desarrollo teórico, aunque recientemente han aparecido implementaciones prácticas simples, por lo que su implementación a corto plazo parece inviable.

Sin embargo, este modelo de computación permite resolver los problemas computacionales más complejos con una pequeña cantidad de recursos y en un

rápido espacio de tiempo. Además, es especialmente bueno cuanto mayor es la complejidad del problema a resolver.

Recientemente se ha demostrado que varias estructuras basadas en *qubits* pertenecen a la clase de circuitos con memoria. Esto ha ampliado el conjunto de *memelementos* con unos nuevos elementos con *respuesta cuántica*, lo que podría traducirse en nuevas aplicaciones para esas estructuras basadas en *qubits*. Este hallazgo podría significar que la *memcomputación* se alimentaría del camino recorrido por la *computación cuántica* y aceleraría su desarrollo, e incluso podría ser la futura alternativa a la *computación cuántica*. [20]

8 – BIBLIOGRAFÍA Y REFERENCIAS

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000.

[2] E. Schrödinger, "The present situation in quantum mechanics". *The science of nature*, vol. 23, pp. 807-812, November 1935.

[3] M. A. Cedano, A. Cedano, J. A. Rubio and A. C. Vega, *Fundamentos de computación para ingenieros*. México D.F.: Editorial Patria, 2014.

[4] G. P. Rédei, *Encyclopedia of Genetics, Genomics, Proteomics and Informatics*. Rotterdam: Springer, 2008.

[5] R. C. Jaeger, *Microelectronic circuit design*. New York: McGraw-Hill, 2011.

[6] A. G. Radwan and M. E. Fouda, *On the Mathematical Modeling of Memristor, Memcapacitor and Meminductor*. Cham: Springer, 2015.

[7] E. Alfonseca, M. Alfonseca and R. Mariyón, *Teoría de autómatas y lenguajes formales*. Madrid: McGraw-Hill, 2007.

[8] S. Arora and B. Barak, Computational Complexity: A Modern Approach. Cambridge: Cambridge University Press, 2009.

[9] A^+ es la conjugada hermítica de la matriz A; $A^+=(A^T)^*$.

[10] D. P. DiVincenzo (2008, February 1). The Physical Implementation of Quantum Computation [Online]. Available: https://arxiv.org/abs/quant-ph/0002077v3

[11] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer". *Proceedings of the Royal Society of London*, vol. 400, pp. 97-117, 1985.

[12] T. Mihara and T. Nishino, "Quantum Computation and NP-Complete Problems", in the Sixth Biennial Conference of the International Society for Augmentative and Alternative Communication, 1997.

[13] M. Di Ventra and Y. V. Pershin (2013, April 8). Memcomputing: a computing paradigm to store and process information on the same physical platform [Online]. Available: https://arxiv.org/abs/1211.4487

[14] M. Di Ventra, Y. V. Pershin and L. O. Chua (2009, January 23). Circuit elements with memory: memristors, memcapacitors and meminductors [Online]. Available: https://arxiv.org/abs/0901.3682v1

[15] Estos cálculos numéricos se han realizado usando el método de integración de Runge-Kutta de cuarto orden.

[16] S. González and D. Arias, “El memristor. Aplicaciones circuitales con amplificadores operacionales”, Escuela de Tecnología Eléctrica, Universidad Tecnológica de Pereira, Pereira, 2014.

[17] F. L. Traversa and M. Di Ventra (2014, November 12). Universal Memcomputing Machines [Online]. Available: https://arxiv.org/abs/1405.0931

[18] Y. V. Pershin and M. Di Ventra (2011, July 16). Solving mazes with memristors: a massively-parallel approach [Online]. Available: https://arxiv.org/abs/1103.0021

[19] El Sistema de ecuaciones de Kirchhoff para el modelo numérico se resuelve utilizando el algoritmo de Coppersmith-Winograd.

[20] S. N. Shevchenko, Y. V. Pershin and F. Nori (2016, July 15). Qubit-based memcapacitors and meminductors [Online]. Available: https://arxiv.org/abs/1602.07230

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR